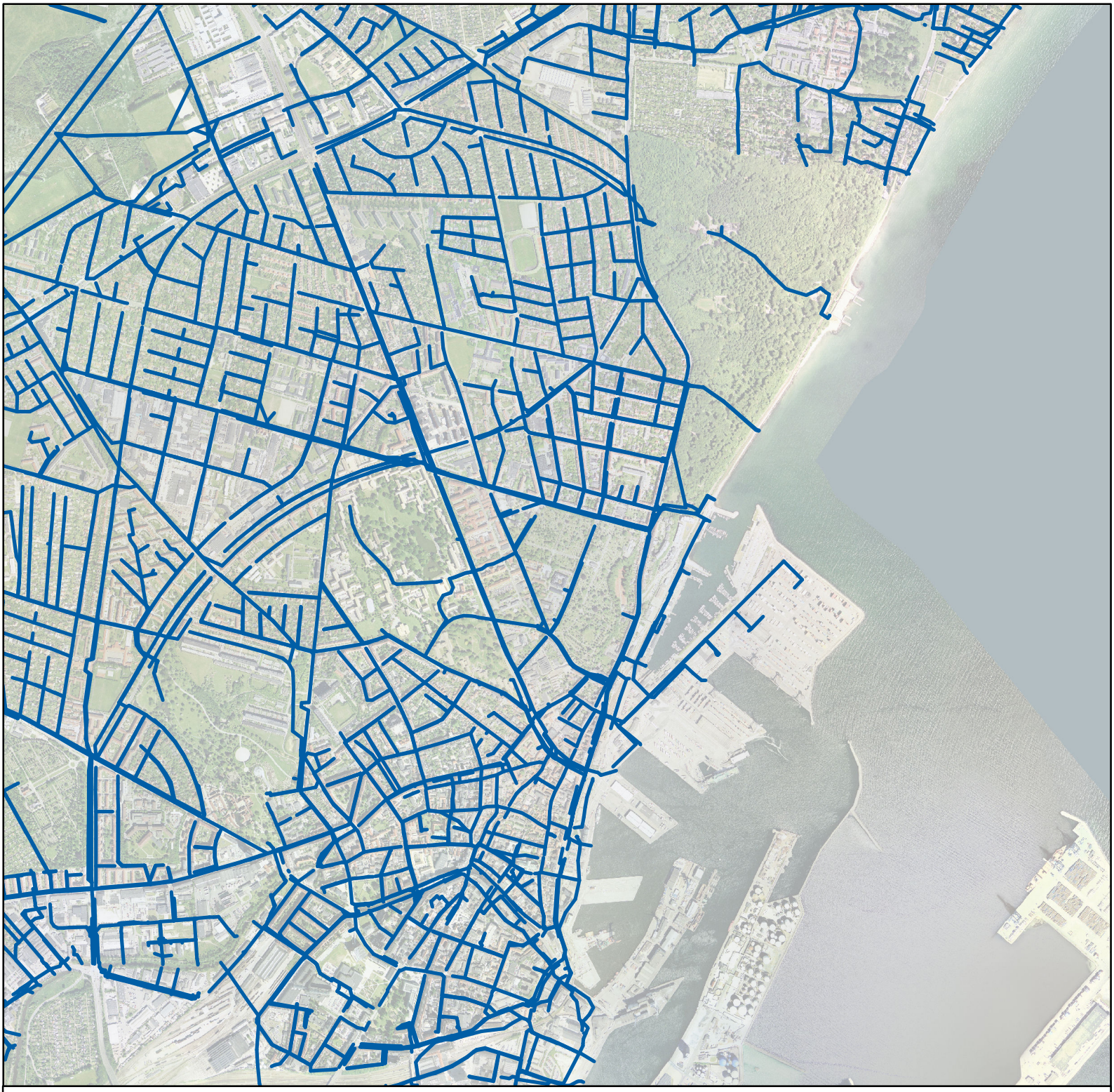
Improving Data for Real-Time Control of Urban Sewer Systems



Master Thesis

Daniel Brødbæk
Aalborg University

2013

Aalborg University

The Faculties of Engineering and Science
Water and Environment
Sohngaardsholmsvej 57
9000 Aalborg
Denmark

Title:
Improving Data for Real-Time Control of Urban Sewer Systems

Subject:
Sensor Validation and Data Assimilation

Project period:
September 2012 - May 2013

Author:

Daniel Brødbæk

Supervisors:
Michael Robdrup Rasmussen
Malte Kristian Skovby Ahm

Copies: 4
Number of pages: 88
Number of appendixes: 38
Ended at: 31/05 - 2013

Abstract:

The purpose of this master thesis has been to improve data for real-time control of urban sewer systems. When real-time control is conducted, it is crucial that decisions are based on reliable information about the state of the sewer system. In order to enhance the reliability of in-situ sensor data, a method capable of conducting a validation in real-time have been developed. The developed method is validated on a case study.

If real-time control is conducted based on a deterministic hydraulic model of a sewer system, then errors on the model prediction will effect the performance. Although in this case the state of the entire sewer system is modelled and can therefore be favourable compared to observations from a finite number of in-situ sensors. In order to get the optimal information about the state of the sewer system the in-situ sensors can be assimilated with a hydraulic model running in real-time. This can be done by the Ensemble Kalman Filter, which calculate the most probable state of the sewer system based on the uncertainty of the measurements and the model prediction. The Ensemble Kalman filter not only update the hydraulic model locally where the in-situ sensors are installed, but the whole model is updated based on the available in-situ sensors. How the Ensemble Kalman filter can be configured for the assimilation is described in this master thesis.

Preface

This master thesis was produced during the 3rd and 4th semester of the Master of Science in Water and Environment, School of Engineering and Science at Aalborg University, in the time period from the 1st of September 2012 to the 31st of May 2013. The themes of the master thesis are sensor validation and data assimilation.

Throughout the report the source reference will be noted with square brackets, with surname of the author and year of the publishing, [surname, year]. The bibliography is at the end of the report.

All figures, tables and equations are numbered according to their chapter, the first figure in chapter four will be numbered 4.1 and the second figure will be numbered 4.2 etc. Every caption contains a source, if no source the figure or table is produced by the writer.

During the master thesis several programmes were developed in MATLAB [Mathworks, 2012]. In the appendix, which can be found in the end of the report, a print of the raw MATLAB codes can be found.

I would like to thank the water company Aarhus Vand A/S, for providing sensors, sensor service and historical data. A special thank to Lene Bassø who was the contact person from Aarhus Vand A/S. Furthermore I would like to thank Mads Uggerby from Envidan A/S for good discussions about the project.

Supervisors on this master thesis were Michael Robdrup Rasmussen and Malte Kristian Skovby Ahm both from Aalborg University.

Aalborg the 31st May 2013

Danish Summary

Emnet for dette afgangsprojekt er sensorvalidering og dataassimilering. Formålet med afgangsprojektet er at forbedre datagrundlaget for realtidsstyring af afløbssystemer.

Realtidsstyring kan være en metode til at imødegå kapacitetesproblemer i afløbssystemer under regn. Ved at anvende realtidsstyring kan udnyttelsesgraden af afløbssystemet forøges. En forudsætning for realtidsstyring er, at informationer om afløbssystemts tilstand er tilgængelige i realtid.

Informationer om afløbssystemets tilstand kan f.eks. opsamles ved at installere in-situ sensorer i afløbssystemet. Dette kunne f.eks. være flowsensorer eller vandstandssensorer. Hvis realtidsstyring foretages på baggrund af sensorer, er det vigtigt, at fejlmålinger identificeres.
Hvis styringen af afløbssystemet er baseret på fejlmålinger, kan det have utilsigtede konsekvenser f.eks. i form af aflastninger til en recipient.

Målinger er dog altid behæftet med en vis usikkerhed, som bør tages i betragtning, når realtidsstyring foretages på baggrund af sensorer. Det kan dog være en fordel, hvis særligt fejlbehæftede målinger kan detekteres og substitueres.

I første del af afgansprojektet præsenteres en generel stepvis metode til at detektere og substituere fejlmålinger fra in-situ sensorer. Metoden bygger primært på at anvende redundante sensorer til at identificere sensorer i fejltilstand. Metoden er blevet testet på et casestudie, der er baseret på målinger fra in-situ sensorer installeret i et afløbssystem i Aarhus. Casestudiet viser, at metoden er effektiv til at identificere fejlmålinger, og ligeledes er metoden effektiv til at substituere fejlmålinger.

I anden del af afgangsprojektet arbejdes der med dataassimilering. Dataassimilering kan anvendes til at kombinere forskellige målinger eller målinger med en prognose, med det formål at få et bedre estimat.

En metode, hvor et Kalman filter anvendes til assimilerer in-situ målinger, præsenteres i andel del. Desuden præsenteres en metode til at assimilere in-situ målinger med en modelprognose af den hydrauliske tilstand i et helt afløbssystem.

En prognose af den hydrauliske tilstand i et afløbssystem kan generes ved hjælp af en hydraulisk model med et regnestimat som input. I forhold til realtidsstyring kan det være en fordel med en modelberegning, da informationer omkring den hydrauliske tilstand dermed er tilgængelig i hele afløbssystemet, dog introduceres der flere usikkerheder på estimatet med modelberegningen. Usikkerhederne på prognosen er en kombination af usikkerheder på regnestimatet, oplandsbeskrivelsen, overflademoddellen og rørmodellen.

Usikkerhederne kan minimeres ved at assimilere in-situ målinger med modelprognosen. Metoden, som præsenteres, er baseret på Ensemble Kalman filteret. Ved at anvende et Ensemble Kalman filter opdateres modelprognosen ikke kun lokalt hvor sensorerne er installeret. Desuden opdateres prognosen under hensyntagen til usikkerhederne på både modelprognosen og in-situ målingerne.

Samlet set giver dette afgangsprojekt forskellige værktøjer, som kan anvendes til at forbedre informationen om tilstanden i et afløbssystem i realtid.

Contents

Introduction 1

The requirements to the water quality in Danish water bodies have been increased during the last decades. The first regulations were introduced in the 1980's in order to reduce the amount of nutrients released.[Environmental Protection Agency, 2012]

In 2000 The EU Water Frame Directive was passed. The directive states that all waters bodies should achieve good conditions in 2015. In Denmark the objective from The EU Water Frame Directive is implemented through the national Water Plans. [Environmental Protection Agency, 2011]

As a central part of the restrictions introduced in the 1980's, the requirements to Waste Water Treatment Plants (WWTP) were increased. However during rainfall events there is a risk that untreated waste water will be discharged to recipients from Combined Sewer Overflow's (CSO's). The release of untreated waste water can have a negative effect on the ecological state of the recipient. As a result the objectives from the Water Plans may not be achieved.

The first modern sewer system in Copenhagen was built in 1859 as a combined sewer system, where waste water and stormwater run-off runs in the same pipe. [Andersen, 2012] CSO's are build in order to regulate the combined sewer system under large rainfall events when the capacity of the system is reached.

In Copenhagen and Denmark in general combined sewer systems are still in use, although a separation of waste water and stormwater run-off have become more widespread in the recent decades. The separation gives different advantages. For instance improved treatment due to more concentrated waste water and less variation in the inflow at WWTP's.

However the separation of the sewer system is expensive, and it will take decades or even centuries to conduct a total separation of the sewer systems in Denmark and in some cases it might not be cost-effective. As an example the vision in Aalborg is to finish the separation in year 2100 [Aalborg Forsyning, 2009].

With the climate changes in mind, an increase in CSO's can be expected if no actions are taken. The Danish Meteorological Institute (DMI), estimate that extreme rainfall events will happen more frequently in the future in Denmark, and according to DMI the climate changes are already taking place [DMI, 2012a].

Besides a separation of the sewer system three different categorise of solutions can be used in order to minimise the discharge of untreated waste water from CSO's, see figure 1.1.

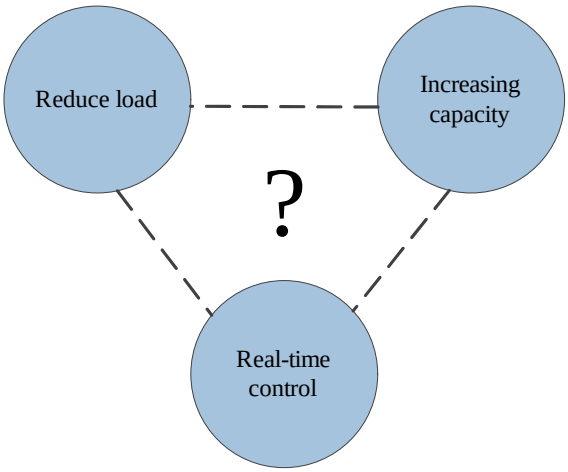


Figure 1.1. Different strategies to minimise CSO's

The traditional way of reducing CSO's, are to increase the capacity of the existing sewer system, or implementing methods to reduce the load of the sewer system. Sustainable Urban Drainage Systems (SUDS) can be used to reduce the load of the sewer system, for instance by implementing green roofs, fascines etc. The last strategy is real-time control (RTC) of the sewer system.

The general idea of RTC is to optimize the performance of the existing system by controlling it in real time. In other words going from a passive sewer system to an active sewer system. The implemented RTC systems can also be useful in separated systems, because there still will be a need to control the hydraulic load to recipients, or the load of waste water to a WWTP. Various articles describe and document the effect of RTC [Stinson, 2005], [Fuchs and Beeneken, 2005] and [Kopecny *et al.*, 2001].

The optimal solution would probably be a combination of increasing the capacity, SUDS and RTC. This master thesis will focus on improving the performance of RTC.

To implement RTC there is a need for information about the state of the sewer system. For instance information about water level or flow rate in real time. The information is needed in order to conduct control decisions. The information can for example be acquired by installing in-situ sensors, or by conducting deterministic model simulations based on precipitation measurements or forecasts. In figure 1.2 the process of RTC is illustrated.

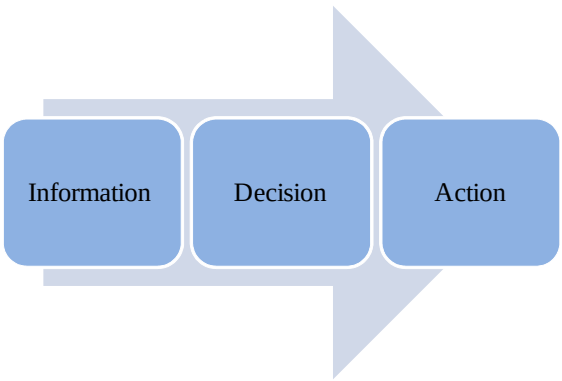


Figure 1.2. RTC process diagram

Based on information about the state of the sewer system a decision is conducted and finally is the decision executed (action). The conducted action can for instance be flow regulation, by a controlled overflow or change in the natural flow direction.

The rate of success for the RTC is dependent on the reliability of the acquired information.

If knowledge about the state of the sewer system is based on a deterministic model simulation, then additional errors from the model is introduced besides the uncertainty related to the precipitation input. However the advantage of controlling the system based on a deterministic model simulation, is that the state of the entire sewer system is modelled.

In-situ sensor measurements are often considered to be the "true value", but there are uncertainties related to all measurements. The accuracy of for example a flow sensor, will depend on the quality of the specific flow sensor, but also fouling and wrong installation could have an influence on the accuracy of the data. During a major hydraulic acquisition program conducted in Australia by Maheepala *et al.* [2001], an expected error of 10 % was found for the used flow sensor, based on laboratory, field and in-situ calibrations. The following monitoring program lasted for three years where a total of 26 urban catchments where monitored for both stormwater run-off and rainfall. The raw flow sensor data from the monitoring program showed some erratic velocity measurements - typically caused by trapped rubbish and debris around the sensors [Maheepala *et al.*, 2001].

As indicated above there is several sources that can cause errors in flow sensor data. Professor Dan Simon from Cleveland State University express it as "*The measurement is like a politician. We can use the information that it presents to a certain extent, but we cannot afford to grant it our total trust.*" [Simon, 2001]

1.1 Problem Statement

This master thesis will aim to answer the following problem statement.

How can the quality of data used for RTC be improved.

1.1.1 Delimitation

The work is separated into two parts.

In the first part of the master thesis an automatic method to validate in-situ sensors in real time is presented. The presented validation method is focusing on how sensor redundancy can be used as a validation tool. As a central part of the developed method, a simple flow routing model have been developed. The flow routing model is capable of estimating the time delay between spatially distributed in-situ sensors. Thereby the redundancy between spatially distributed in-situ sensors can be monitored.

The second part of the master thesis deals with data assimilation. A central part of the second part is the Kalman filter and how it can be used to improve the knowledge about the true state of a sewer system. A method to assimilate in-situ sensor measurements are presented in the second part. Furthermore is a method to assimilate in-situ sensors with a deterministic flow model presented.

The Principle of a Sensor 2

The understanding of the operating principles of sensors are important, in order to understand the reason for the errors associated to sensor measurements. First some definitions are introduced.

- Precision. A sensor is precise, if the noise on its measurements are low
- Accuracy. A sensor is accurate, if it on the average measure a value close to the true value

For instance, if it is assumed that a sample of water level measurements are present in a channel with constant water level, then it can be explained by the following scenario.

If the average of the sample are close to the true value, then the sensor is accurate, and if the difference between the sample members are low, then the sensor is also precise.

The accuracy of an inaccurate sensor can improved by continuously calibration. Different events are responsible for imprecise measurements. The influence of some events can be minimised if precautions are taken. For instance a location with very unstable flow conditions can effect the precision of measurements from a sewer system in a negative direction.

Other error sources are related to the specific type of sensor. Usually are sensors in urban sewer systems, used to monitor flow rate, water depth, temperature etc. In order to measure these quantities, different methods can be used. Each method are associated with different uncertainties, however all methods conduct an indirect measurement of the quantity of interest.

For instance could temperature be measured indirectly, by monitoring the electrical resistance in a metal. The resistance in the metal could again be measured indirectly, by measuring the voltage drop over the resistance. Then to estimate the temperature it would require a preliminary calibration, in order to establish a correlation between the measured voltage drop and the temperature. The precision of the sensor will be depending on the correlation.

An additional error is introduced when data is stored. This is because a computer cannot handle a continuous signal, it need to receive discrete values. Therefore an analogue-digital-converter (A/D-converter) is used, to transform the signal from continuous to discrete. Depending on the bits of the A/D-converter, a finite number of possible output signals will be present. Therefore with a increasing number of bits the error will decrease. Furthermore is the sample rate also of importance. Because the uncertainty, of values in between the discrete values, will increase with a decreasing sample rate. The sample rate is normally given in hertz (s^{-1}). In figure 2.1 the process is sketched.

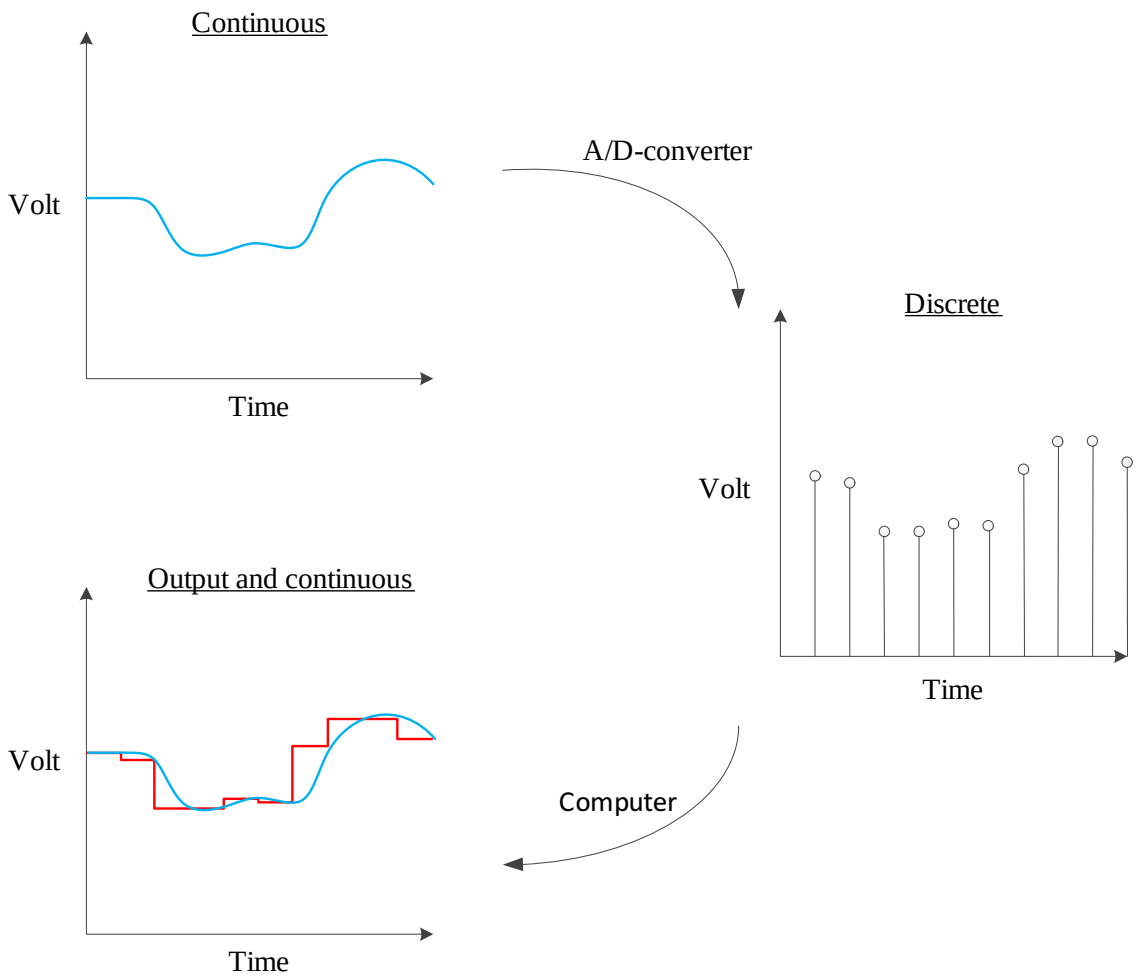


Figure 2.1. An example of the error introduced, when a signal is transformed from a continuous analogue signal to a discrete digital signal

As described in this chapter, errors are introduced as a consequence of different events. Precautions can be taken to minimise errors, however sensor measurements will always be subject to some errors. Besides the described factors which can effect the precision, then other events such as power failure or blocking of a signal, can result in a missing signal or suddenly changes in the signal.

Part I

Sensor Validation

The first part of this master thesis presents an effective method for validation of data acquired by in-situ sensors in sewer systems. The presented method is capable of identifying errors and replacing them in real-time.

The developed method is tested on a case study based on in-situ sensor measurements from Aarhus.

The Validation Method 3

Previously studies have been conducted in order to develop methods for validation of hydraulic data from urban sewer systems. For instance see [Bertrand-Krajewski *et al.*, 2003], [Branisavljevic *et al.*, 2010], [Mourad and Bertrand-Krajewski, 2002], [H.J.Liefting and Langeveld, 2008] and [Bijnen and Korving, 2008].

Sun *et al.* [2011] have conducted a study of previously developed techniques in order to identify areas which still need attention. They point out that material/analytical redundancy for data validation is still a potential research area. Material redundancy refer to two sensors measuring the same quantity which are redundant at a given site, while analytical redundancy refers to two sensors measuring quantities which are correlated at a given site.

Based on the work conducted by Sun *et al.* [2011] the validation method is developed with a focus on how redundancy can be used as validation tool.

As a part of the developed validation method a simple flow routing model have been developed for open channel flows. The developed flow model is capable of estimating the time delay between sensors. In chapter 4 the flow routing model is documented.

3.1 Validation Procedure

In figure 3.1 the flow of the developed validation procedure is given.

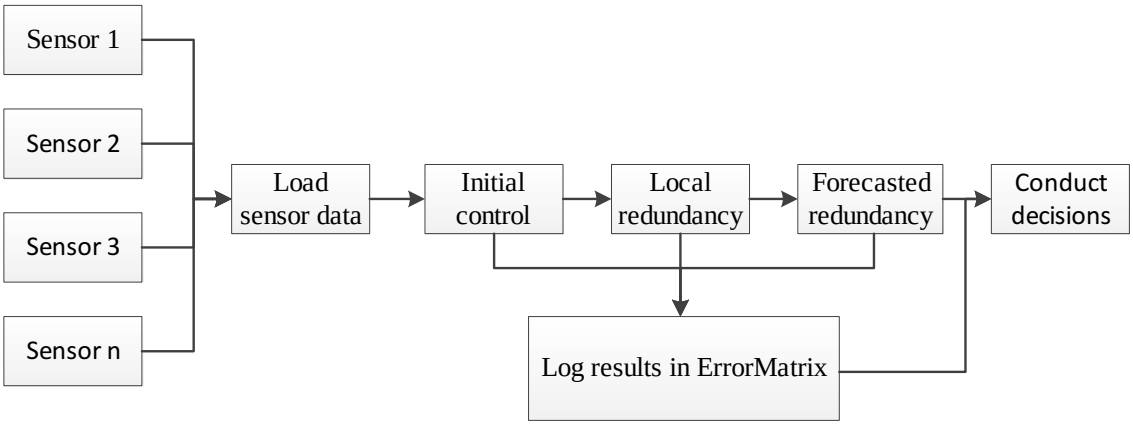


Figure 3.1. The flowchart of the validation princip

The validation method is set up as a stepwise procedure as presented in the flowchart in figure 3.1. For a given system *n* sensors are available, and in the first part data from all sensors are loaded.

Then all loaded data are subject to an initial control. The performance of the initial control is logged in a matrix named "ErrorMatrix". Afterwards two redundancy tests are applied and the performance of these test are again logged in the "ErrorMatrix".

In the last step of the flowchart the results logged in the "ErrorMatrix" are loaded, and decisions are conducted based on the loaded results. Decisions are conducted based on predefined logical statements evaluating the response of the individual tests.

The last four steps in the validation procedure will be described in the following subsections.

3.1.1 Initial Control

In this step all sensor data are subject to three simple tests. The test are launched in order to identify errors caused by external sources. For instance blocking of signal and power failure. The three tests are given below.

1. Does data exist
2. Test for non physical value
3. Test for zero

If data does not exist it could be due to different reasons. For instance the sensor is in off mode, connection failure or the sensor failed to conduct a measurement. A non physical value should be identified according to a certain predefined threshold. The non physical value can both be negative and positive. The threshold value should be set specific for each sensor depending on the sensor and the location of the sensor. Testing for zero can help identifying errors in measurements.

3.1.2 Local Redundancy Test

This part is only conducted if two redundant sensors are located at the same site in the sewer system. In this report it will be referred to as local redundant sensors. Furthermore both sensors should have passed the evaluation of the initial control.

The test does not distinguish between material and analytical redundancy. If it is not the same quantity which is measured then the measurements are transformed to represent the same quantity.

The residual between the measurements will be calculated in order to identify abnormal residuals, and to identify if sensors are gliding. The residual is calculated as given in equation 3.1.

$$\varepsilon = S_1 - S_2 \tag{3.1}$$

S	Sensor measurements	[-]

Abnormal residuals are identified according to a threshold. The threshold is calculated at each time step based on a predefined acceptable percentage deviation see equation 3.2.

$$T = \frac{S_1 \cdot P}{100} \tag{3.2}$$

P	Acceptable percentage deviation	[%]

If the absolute value of the residual is larger than the threshold, it will be logged in the "ErrorMatrix". The advantage off not using a fixed threshold, is that small deviations coming from gliding zero values can then be detected as well as single outliers.

The threshold is calculated with respect to the sensor which should be validated.

If for instance an area velocity flow sensor is installed, then the actual measurements are velocity and pressure and the flow rate is calculated. Several different redundancy tests can be performed in this case however it decided to monitor the residual between the output flow rate and a flow rate estimated based on the pressure measurement. This require that a Q-h-relation have been calibrated.

3.1.3 Forecasted Redundancy Test

In this step the redundancy between spatially distributed sensors are monitored. For this purpose a flow routing model have been developed. The developed flow routing model estimate the time delay between sensors in order to compare there measurements. The flow routing model is documented in chapter 4. The test used in the local redundancy test will also be used in this test in order to identify outliers.

3.1.4 Conduct Decisions

Finally when all tests are finished decisions are automatically conducted based on the performance of each test. As previously mentioned the performance have been logged in the "ErrorMatrix". The number of rows in the matrix will correspond to the number of sensors while the number of columns will be five - one for each test. Three test was conducted in the initial control, one local redundancy test and one forecasted redundancy test.

The response of the tests are evaluated based on logical statements. The logical statements is visualised in a decision tree in figure 3.2.

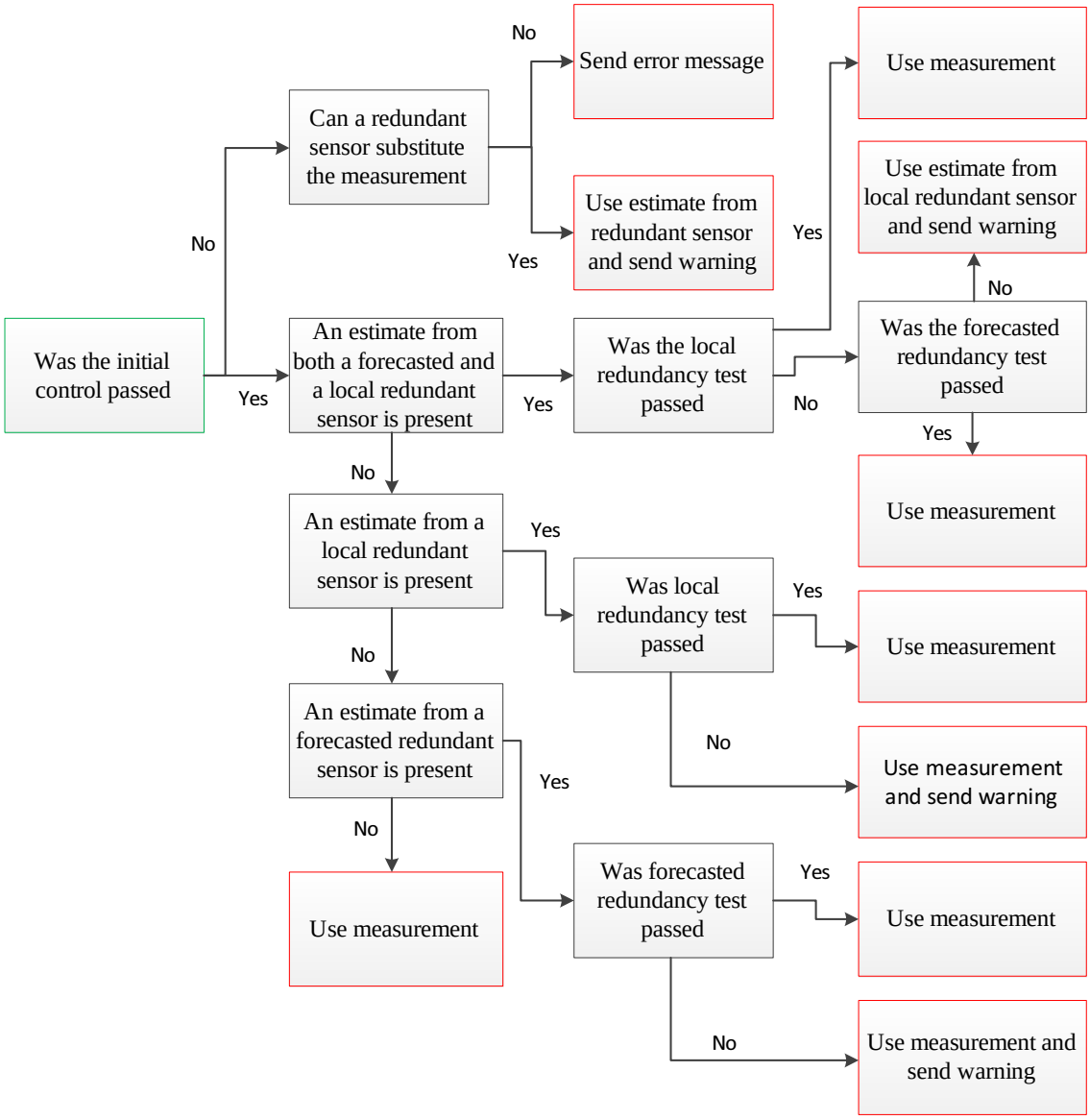


Figure 3.2. Logical statements used by program to conduct decisions, here represented in decision tree

The starting point in the decision tree is marked by a green frame and the possible endpoints are marked by red frames.

Some times a warning is send together with the outcome. This is done if the program fails to establish correlation between the majority of redundant sensors.

In order to visualise the "ErrorMatrix" and how the presented validation method will respond to the results logged in the matrix, a theoretical example is examined the next section.

3.2 Theoretical Example

Consider the scenario in figure 3.3.

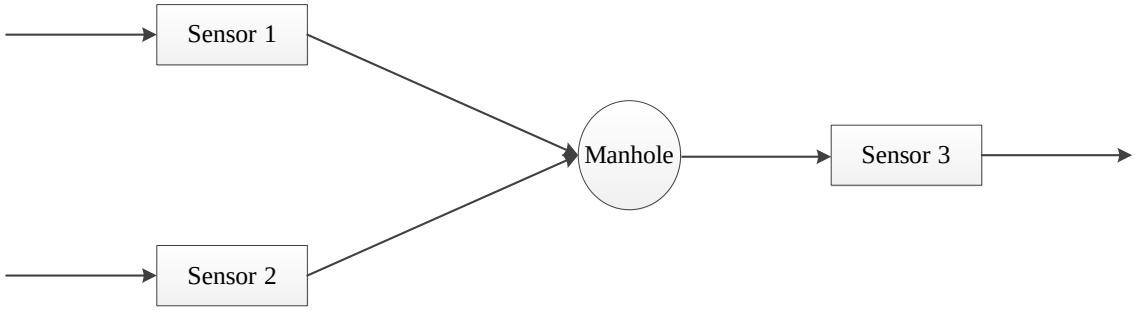


Figure 3.3. Simple flow scenario

In figure 3.3 the flow direction is from sensor 1 and sensor 2 towards sensor 3. If it is assumed that sensor 1 and 2 both are water level sensors and sensor 3 is a flow sensor, then the "ErrorMatrix" at a given time step could be as given in the table below.

Sensors	Data existence	Physical value	Zero	Local redundancy	Forecasted redundancy
1 (Water level)	1	1	1	NaN	NaN
2 (Water level)	1	1	1	NaN	NaN
3.1 (flow)	1	1	1	1	1
3.2 (pressure)	1	1	1	NaN	NaN
3.3 (Velocity)	1	1	1	NaN	NaN

Table 3.1. Example of the "ErrorMatrix". 1=Passed, 0=Failed, NaN=Not performed

A total of five sensors are present since it is assumed that the flow sensor measure the pressure, the velocity and estimates the flow.

Initially at each time step all entrances are set to one in the matrix meaning the control is passed. However if a test is failed one is substituted with a zero. In case a test is not performed one is replaced by NaN, a syntax indicating it is not a number.

In the example in table 3.1 it can be seen that all sensors passed the initial tests. No local redundancy test have been performed at the water level sensors since the water level sensors only measure one quantity. Furthermore have no forecasted redundancy test been performed since there is no sensors upstream sensor 1 and 2.

At the flow sensor a local redundancy test have been performed and it have been passed. This means that the flow estimate, based entirely on the pressure sensor, does not deviates more than accepted from the output flow rate. Furthermore have the forecasted redundancy test been passed. This means that the combined estimate of the flow rate at sensor 3, based on the water level measured at sensor 1 and 2, does not deviates more than accepted from the output from the flow sensor.

No redundancy tests have been performed with respect to the pressure and velocity measured at sensor 3. This is because the flow rate, is the target for the redundancy tests at the given location.

If the decision tree again is studied then the final outcome from the validation procedure can be visualised. The blue arrows correspond to how the flow output from the flow sensor would be evaluated.

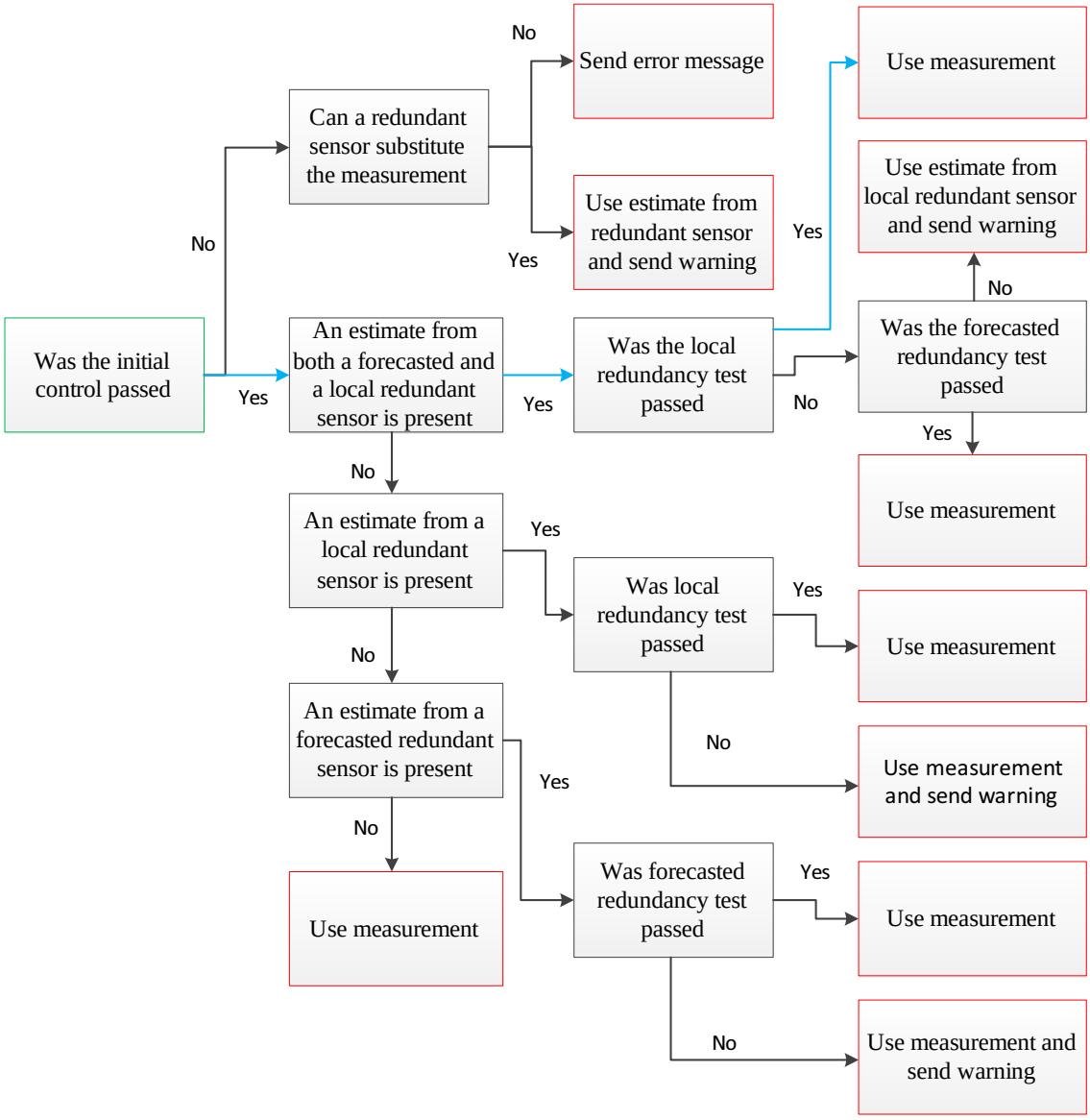


Figure 3.4. Evaluation of the flow measurement

3.3 Summary

In this chapter a validation procedure have been presented. Before the validation procedure is applied to a specific area a pre-analysis have to be conducted. In the pre-analysis relations between sensors have to be established, and some parameters have to be set. For instance thresholds to detect non physical values in the initial control.

The developed flow routing method, which is presented in the next chapter, also needs to be calibrated for the specific area.

Flow Routing Method 4

The Flow routing model was developed in order to make spatially distributed sensors redundant.

Under normal circumstances in a sewer system pipes are partially full, and the flow can therefore be considered as a open channel flow. The method have therefore been developed in order to estimate the time delay, under the assumption of a free water surface. In case of full running pipes the model can therefore not be directly applied.

In order to evaluate how the system should response in case of full running pipes, pre-analysis of the specific sewer system could be conducted with a numerical model of the sewer system.

4.1 Development of the Model

In order to develop the method a theoretical example was studied see figure 4.1.

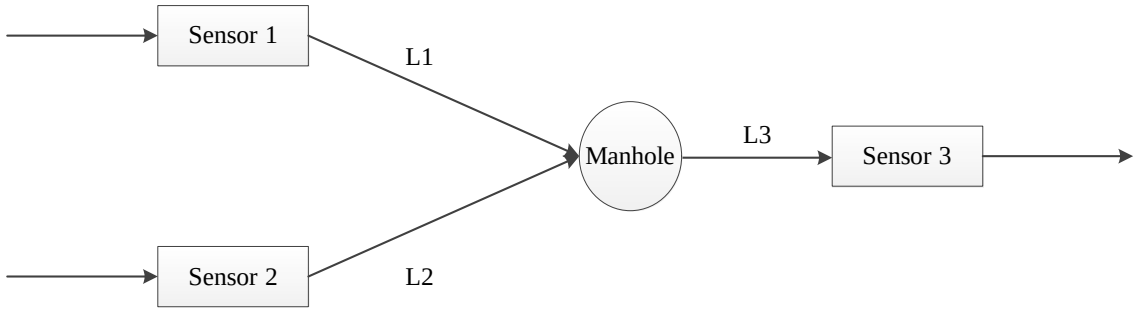


Figure 4.1. Simple flow scenario

The set up is identical to the example studied in chapter 3.

The developed method is based on the assumption, that a flow rate measured upstream a given sensor will at later time be present at the sensor, if there is no sources or sinks in between. This means that the flow is considered to be quasi stationary and uniform between the sensors.

Therefore instead of directly forecasting measurements from sensor 1 and 2 the delay between the sensors is estimated. However should the delay be varying based on the measurements or is a constant delay a good assumption?

The average velocity is depending on the water level and the gradient of the energy level. Furthermore if waves is present the delay will decrease further due to the wave celerity. [Brorsen and Larsen, 2009] However it might give reasonable results to a assume a constant delay.

In order to investigate this a simple model of the system visualised in figure 4.1 was set up in MIKE URBAN [DHI, 2011]. Then different flow rates were applied as boundary condition at sensor 1, and the travel time to sensor 3 was estimated at each applied boundary condition.

The theoretical study revealed, that the delay varies to much to accept the assumption of a constant delay. The theoretical study can be seen in appendix A.

If constant delays cannot be used, then the delays have to be estimated based on the measurements in real time. However the delays cannot be estimated based on measurements acquired from sensor 3, since it then will be impossible to replace data from the sensor in case of errors. Therefore the delay should entirely be estimated based on measurements upstream the sensor of interest.

If mass conservation is assumed, then an estimate of the flow rate at sensor 3, based on sensor 1 and 2, see figure 4.1 can be calculated by equation 4.1.

$$Q_3(t) = Q_1(t - \tau_1^{'}(t)) + Q_2(t - \tau_2^{'}(t)) + \varepsilon \tag{4.1}$$

Q_3	Flow at sensor 3	$[m^3/s]$
Q_1	Flow at sensor 1	$[m^3/s]$
Q_2	Flow at sensor 2	$[m^3/s]$
t	Time index	[s]
$\tau_1^{'}$	Total delay from sensor 1 to 3	[s]
$\tau_2^{'}$	Total delay from sensor 2 to 3	[s]
ε	Residual	$[m^3/s]$

The residual in equation 4.1 is due to different reasons. Noise on measurements and errors related to the assumption that the flow rate will not change through the system. In order to give the best estimate the residual should be minimized as much as possible, this is done by estimating the correct delays.

The delay at sensor 1 and sensor 2 are depended of each other since the delay from the manhole to sensor 3 are the same. The delays in equation 4.1 could be expressed as.

$$\begin{aligned} \tau_1^{'} &= \tau_1 + \tau_3 \\ \tau_2^{'} &= \tau_2 + \tau_3 \end{aligned} \tag{4.2}$$

τ_1	Delay in link between sensor 1 and manhole	[s]
τ_2	Delay in link between sensor 2 and manhole	[s]
τ_3	Delay in link between manhole and sensor 3	[s]

Due to the assumption of quasi stationary and uniform flow the delays τ_1 and τ_2 can be estimated by iteration of equation 4.3.

$$\frac{L_1}{U_1(t-\tau_1)\tau_1} = 1 \qquad \frac{L_2}{U_2(t-\tau_2)\tau_2} = 1 \tag{4.3}$$

L_1	Length of link between sensor 1 and manhole	[m]
$U_1(t)$	Average velocity in link between sensor 1 and manhole	[m/s]
L_2	Length of link between sensor 2 and manhole	[m]
$U_2(t)$	Average velocity in link between sensor 2 and manhole	[m/s]

The iteration of τ_1 and τ_2 in formula 4.3 can be conducted by increasing the delays until the conditions is full filled. Due to the variation of the average velocities U_1 and U_2 in time these have to be calculated at each time step.

The average velocities U_1 and U_2 can be estimated if the water depth is known since the flow is assumed to be quasi stationary and uniform.

After the delays have been estimated from equation 4.3, then the flow rate at time t in the manhole between the sensors can be estimated by equation 4.4.

$$Q_m(t) = Q_1(t - \tau_1(t)) + Q_2(t - \tau_2(t)) \tag{4.4}$$

When the flow rate at the manhole is calculated then the flow rate at sensor 3 can be estimated by the same procedure. However the water depth have to be iterated in order to calculate the velocity in the partly full pipe.

For the iteration procedure Newtons method is used. Newtons method use the derivative of the function to estimate the next guess in the iteration. The derivative of $f(x_n)$ can be calculated by equation 4.5.

$$f'(x_n) = \frac{\Delta y}{\Delta x} = \frac{f(x_n) - f(x_{n+1})}{x_n - x_{n+1}} \tag{4.5}$$

x	Variable to be iterated	$[-]$
n	Iteration index	$[-]$
$f(x)$	Function	$[-]$
$f'(x)$	The derivative of the function	$[-]$
Δy	Change in function value	$[-]$
Δx	Change in variable to be iterated	$[-]$

[Edwards and Penney, 2008]

If $f(x_{n+1})$ is substituted with zero and the equation is rearranged then the next guess in the iteration procedure can be estimated from equation 4.6.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.6}$$

[Edwards and Penney, 2008]

If Mannings equation is used to establish a correlation between flow rate and water depth then $f(x)$ would be as presented in equation 4.7.

$$f(x) = f(h) = m\, R(h)^{2/3}\, \sqrt{I_0}\, A(h) - Q = 0 \tag{4.7}$$

m	Mannings number	$[\mathrm{m}^{1/3}/\mathrm{s}]$
R	Hydraulic radius	$[\mathrm{m}]$
I_0	Bottom slope	$[\mathrm{m}/\mathrm{m}]$
A	Flow area	$[\mathrm{m}^2]$
Q	The flow rate	$[\mathrm{m}^3/\mathrm{s}]$
h	Water depth	$[\mathrm{m}]$

However precautions should be taken if Mannings equation is used in circular pipes nearby full running. Because Mannings equation predict the maximum flow rate to be shortly before the pipe is full running [Winther *et al.*, 2006]. This can make the iteration procedure more complicated.

4.2 General Formulation of the Flow Routing Model

The model can be used to forecast an arbitrary number of measurements converging in downstream pipes. If N sensors are present the general form of equation 4.1 would be as given in equation 4.8.

$$Q_N(t) = \sum_{n=1}^{N-1} (Q_n(t - \tau_n^{'})) + \varepsilon \tag{4.8}$$

N	Total number of sensors	[-]
n	Index referring to individual sensors	[-]
$\tau_n^{'}$	Delay from sensor n to sensor N	[-]

The delays in each link is stepwise calculated through the system, and finally the total delay at each sensor can be calculated. As a side bonus of the model, time series of the flow rate in all manholes between input sensors will be estimated.

4.3 Summary

In this chapter a method to estimate the delays between spatially distributed sensors have been presented.

Before the model is directly applied to a certain area, it is important to conduct a pre-analysis of assembled data. This should be done to calibrate Q-h-relations and maybe other functions which can transfer a specific measurement to a flow rate.

If the method is used to forecast flow rates over long distances it might be necessary to apply source or sink terms.

In the next chapter the developed validation method and the flow routing method will be tested on data from a case study.

Case Study 5

The case study was conducted in order to evaluate the performance of the developed flow routing method and the validation method on real data.

A sensor set up similar to the one studied in the development of the methods were examined, i.e. a location where two pipes are converging. The specific location for the sensors was chosen in order to minimise disturbance. Disturbance can be introduced by different circumstances.

For instance by physical changes near the sensors or if the flow changes between sub critical and supercritical flow. In order to minimise the impact on the sensors precision, caused by disturbances in the flow, the following guidelines was followed when the exact location for the sensors were determined.

- The pipes should be straight and the diameter should not be changing near the sensors
- No side runs near the sensors
- Froudes number should be less than one for all water depths at the sensors

Froudes number should be less than one, since one is the transition value between subcritical and supercritical flow. [Brorsen and Larsen, 2009]

The water company Aarhus Vand A/S provided the sensors and were responsible of maintenance during the measurement campaign and therefore the sensors were installed in the sewer system in Aarhus.

It turned out to be more difficult than first expected to find a location for the sensors which could fulfil the defined criterias. In general the problem was Froudes number. If Froudes number should be less than one for all water depths, it require the slopes to be low for the pipes at the sensors. Although an appropriate location was found. The measurements were conducted in the autumn 2012.

5.1 The Catchment

Aarhus is a city situated in the central part of Jutland in Denmark and the case catchment is located in the northern part of the city see figure 5.1.

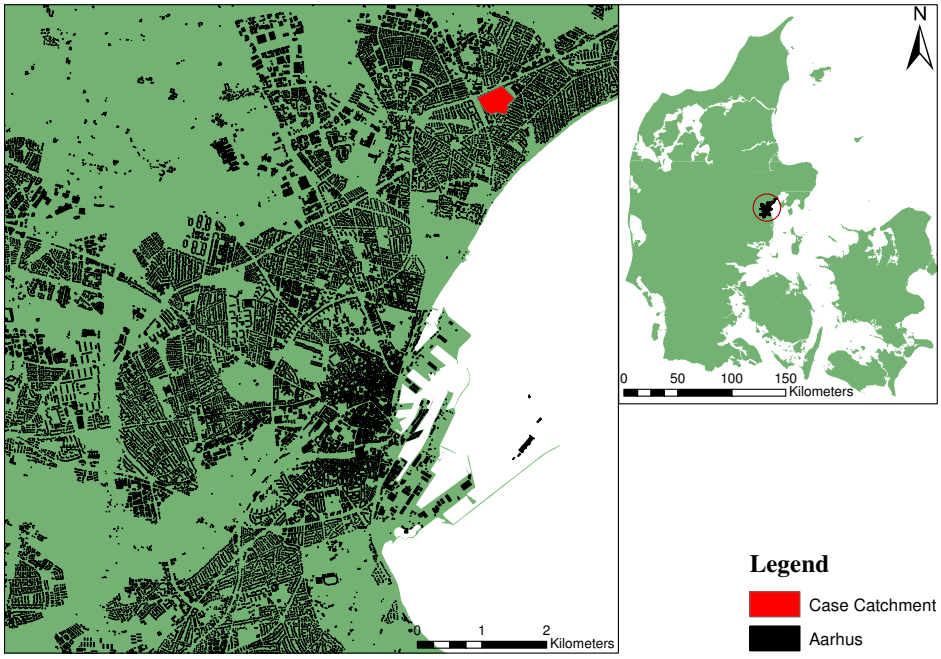


Figure 5.1. The location of the case study catchment in Aarhus

Figure 5.2 gives a closer view of the catchment.

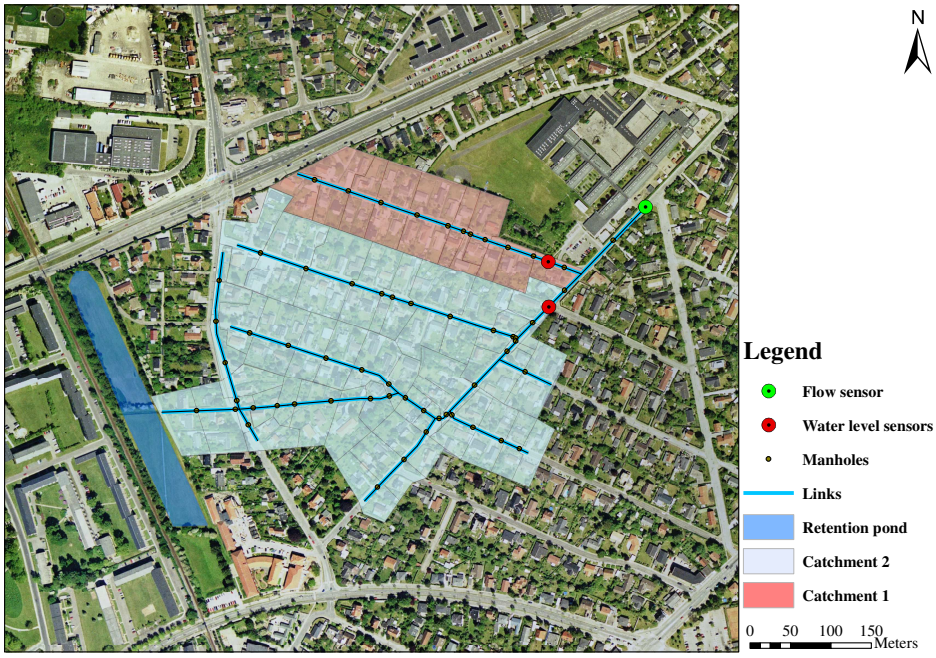


Figure 5.2. The catchment for the case study and the set up of the sensors

The flow direction in the sewer system is from south west towards the sensors. The sewer system is separated, and the sensors was mounted in the rain system.

The catchment area can be divided into two sub catchments - one sub catchment for each water level sensor as visualised in figure 5.2. Both catchments consist of residential areas. The catchments are of different size, the smallest named catchment 1 in figure 5.2 has approx a total catchment area of 3 ha, and catchment 2 has a total catchment area of approx 12 ha.

An open retention pond is located south west of the case catchment see figure 5.2. The case catchment is not discharging to the retention pond, but the catchment south west of the retention pond discharge to the retention pond in case of a larger rainfall event.

Under normal circumstances the retention pond does not interfere with the case catchment, although if the water rises to a certain level in the retention pond, then the retention pond will discharge to the sewer system in the case catchment.

In figure 5.3 a closer view of the sensor setup is given.

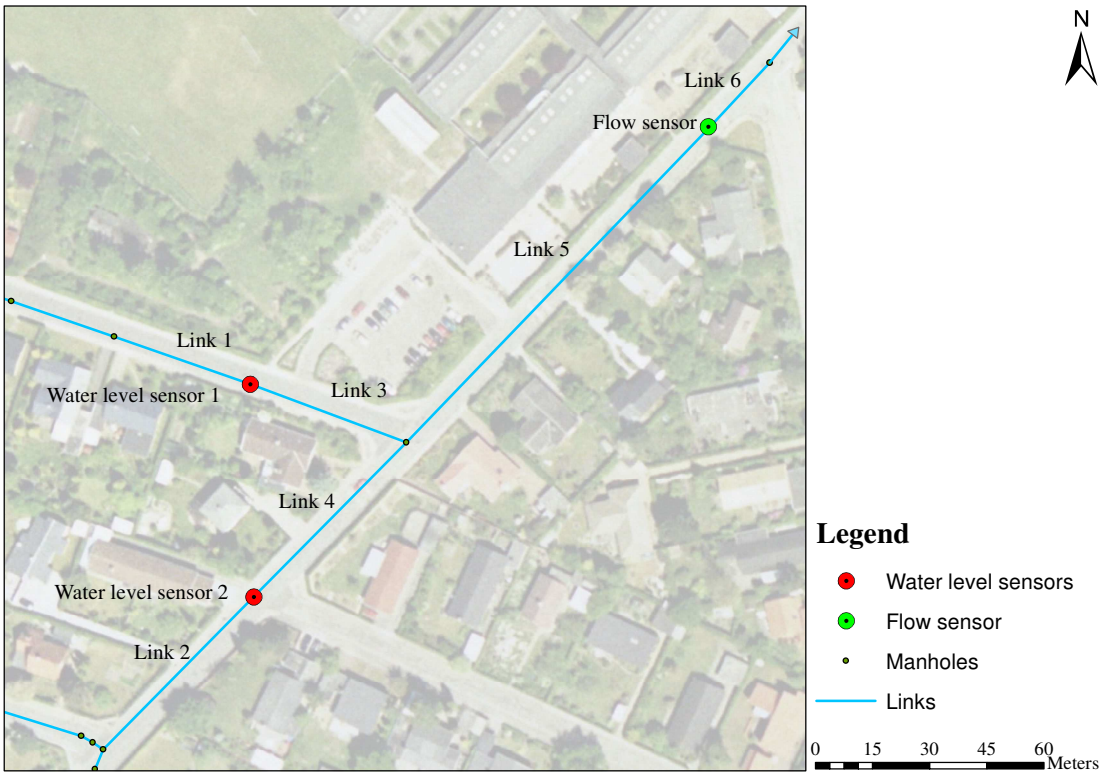


Figure 5.3. Setup of the sensors

Information regarding the links in figure 5.3 are given in table 5.1.

Link	Type	Slope [‰]	Diameter [mm]	Length [m]
1	Concrete	3	350	38
2	Concrete	1	600	56
3	Concrete	4	350	43
4	Concrete	2	600	57
5	Concrete	1	600	114
6	Concrete	0	600	24

Table 5.1. Information about links at the sensors

5.2 Installed Sensors

5.2.1 Water level sensors

The sensors named water level sensor 1 and 2 in figure 5.3 are the same type of ultrasonic sensors. The sensor type measure the distance from the sensor head to the water surface by ultrasonic sound as visualised in figure 5.4.

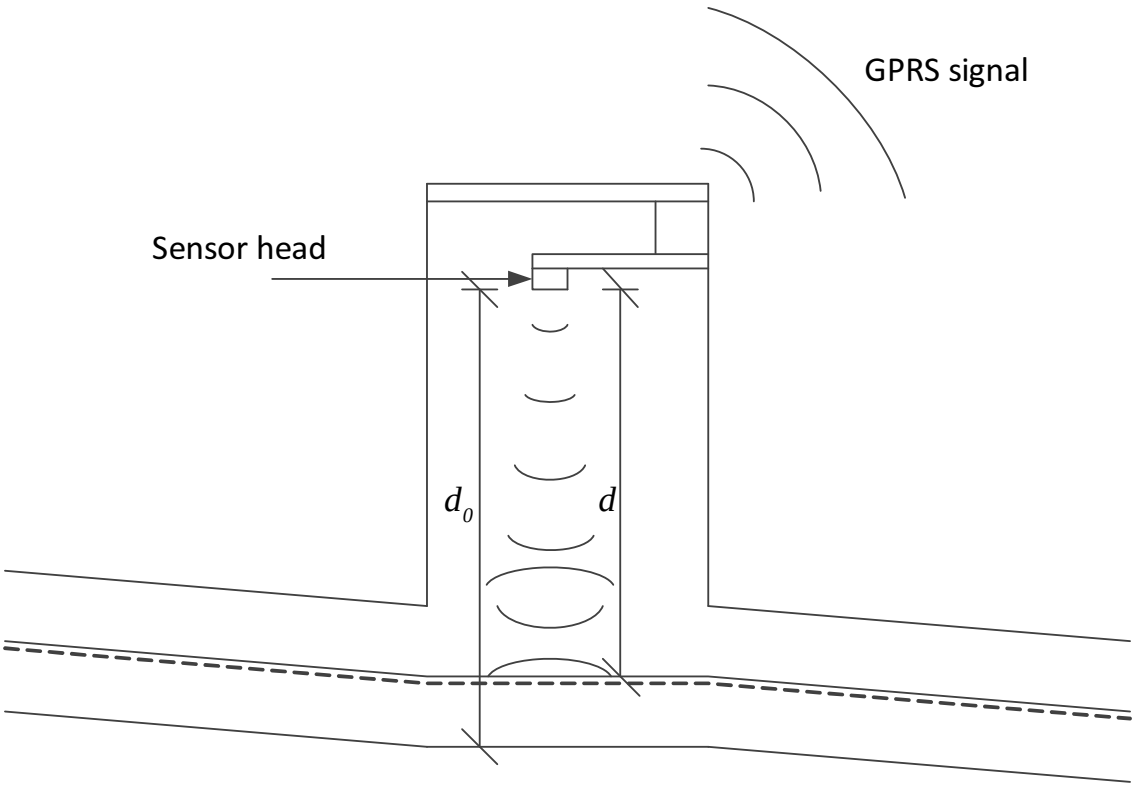


Figure 5.4. Principle sketch visualising the set up of water level sensor 1 and 2

By measuring the time before the signal is returned the distance d can be calculated, since the pulse is travelling with the speed of sound. The water depth can then be calculated by subtracting d from d_0. The data is extracted from the sensor automatically by an external computer via the GPRS network.

The mounting of water level sensor 1 and 2 are shown in figure 5.5 and 5.6 respectively.

Figure 5.5. The mounting of water level sensor 1 in the manhole.

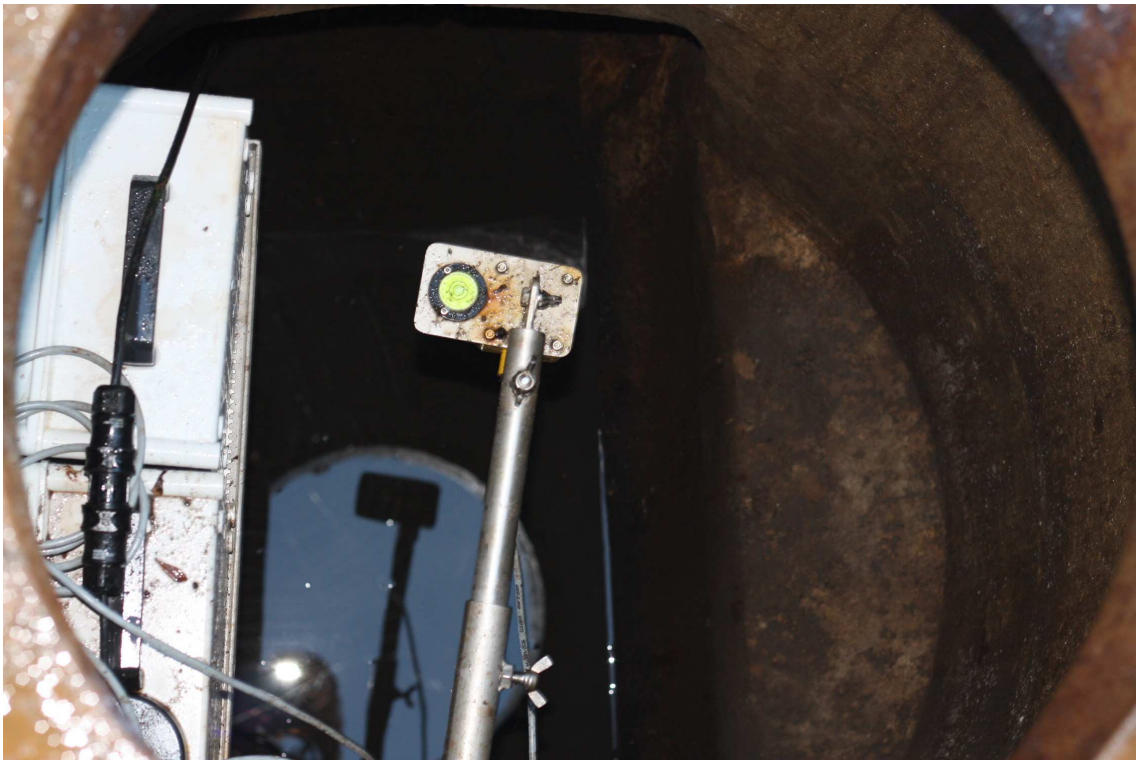


Figure 5.6. The mounting of water level sensor 2 in the manhole.

The distance d_0 at both sensors is given in table 5.2.

Sensor	Distance d_0 [mm]
Water level sensor 1	550
Water level sensor 2	990

Table 5.2. Distance from water level sensors to bottom of manholes

During the measurement period sedimentation of sand have been observed at water level sensor 2. Furthermore a dry weather flow have been observed at water level sensor 2, while at water level sensor 1 no dry weather flow have been observed.

Besides the uncertainties related to the flow conditions at the location where the sensors were installed, there are also different uncertainties related to the concept of this specific type of sensor.

- The speed of sound is influenced by temperature, although the sensors are compensating for this, meaning the error is aprrox 0.02 % of the measured distance.
- Error due to linearity when transferring output signal to distance is appox 0.2 % of measuring span. The measuring span of the sensors is set to 0.25 - 6 meters, meaning that the maximum error is approx 12 mm due to linearity.

The sensors was set to measure the water depth in a temporal resolution of 1 minute. A detailed description of the ultrasonic sensors can be seen in the manufactures manual. [Banner Engineering, 2012].

5.2.2 Flow Sensor

The flow rate was measured by an area velocity sensor submerged in the link just upstream the manhole. The sensor measures the pressure by a pressure transducer and the velocity is measured by ultrasonic sound. By measuring the shift in frequency from reflection of particles in the water the velocity is estimated.

The flow rate is then calculated by integrating the velocity over the flow area. The set up of the sensor is visualised in figure 5.7 and figure 5.8 shows a picture of the mounted data logger.

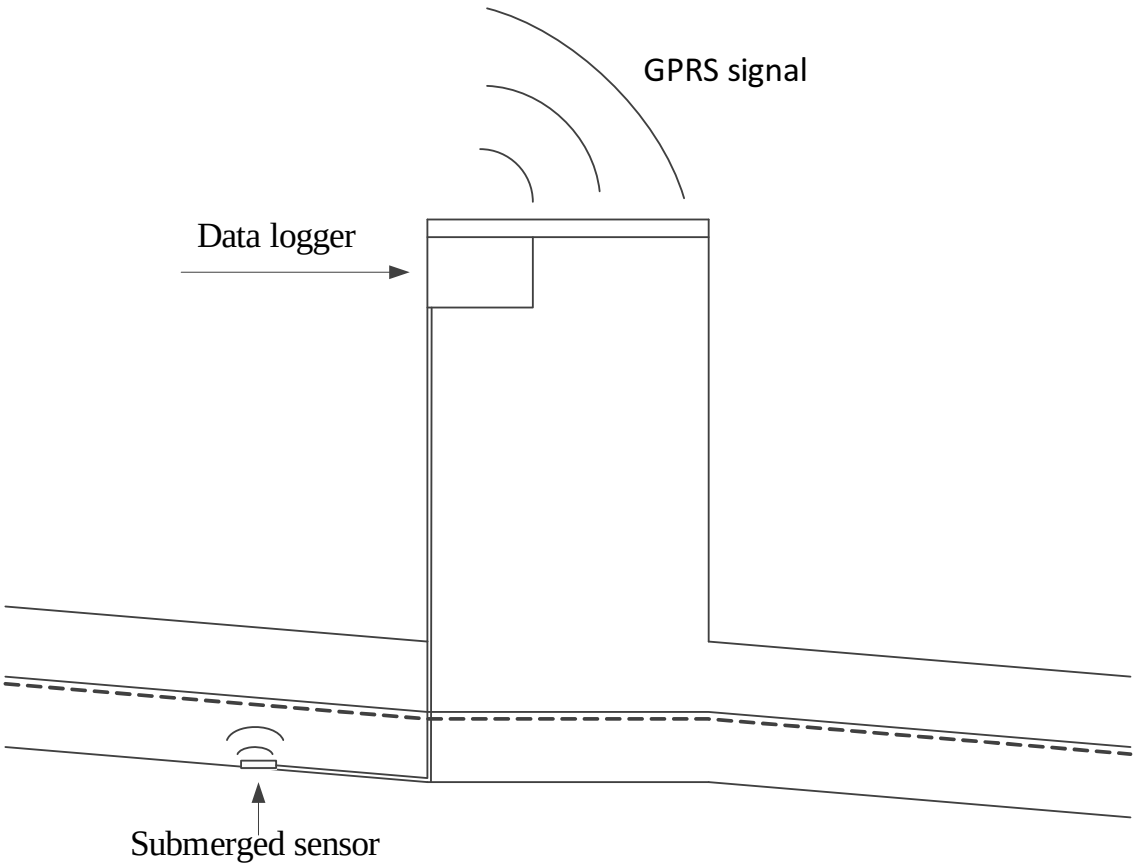


Figure 5.7. Principle sketch visualising the set up of the flow sensor

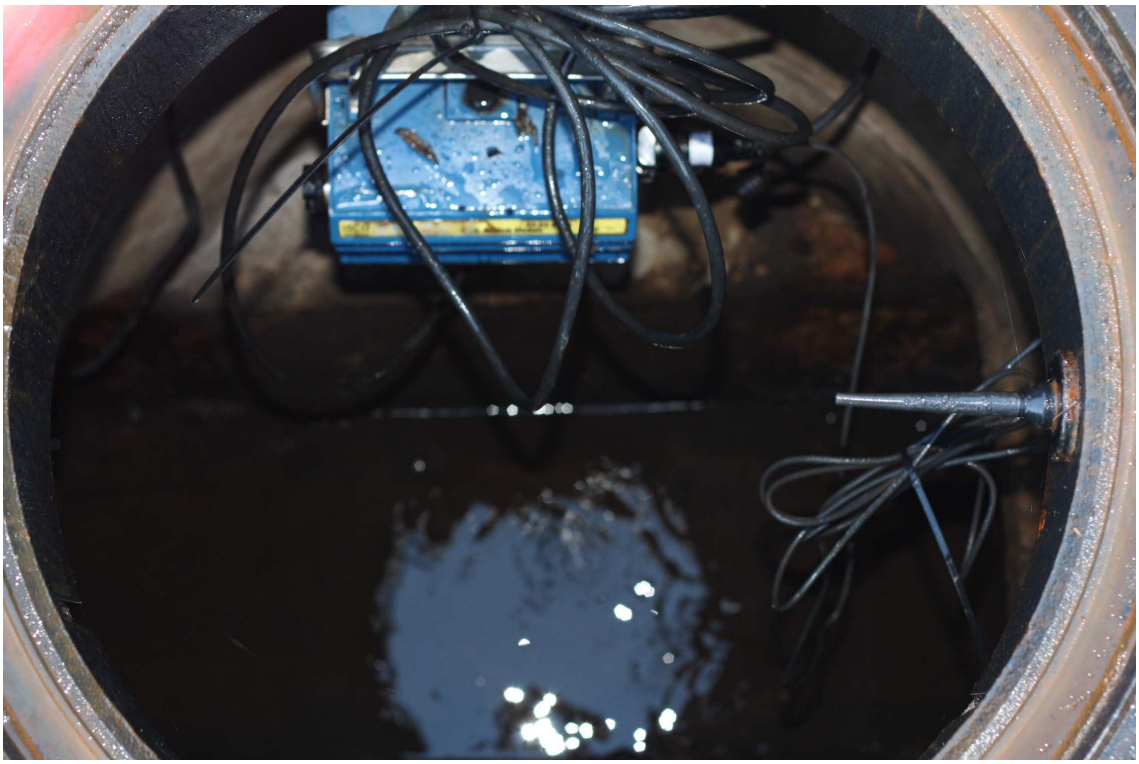


Figure 5.8. Data logger in the manhole where the flow sensor (not visible) is mounted

Sedimentation of sand and a dry weather flow were observed at the flow sensor. The sedimentation depth were measured to approx 30 mm at each inspection. Sedimentation of sand might influence the measurement, if the sediments block the ultra sonic sound use to measure the velocity.

During installation of the sensor a sedimentation depth of 30 mm was defined in the software of the flow sensor, by the crew from Aarhus Vand installing the sensor. Thereby the sensor should be able to take the reduced flow area into account when estimating the flow.

In order to prevent sedimentation from blocking the signal from the sensor, the sensor was not mounted in the bottom of the pipe, but a little up the side so the sensor was above the sediments.

There are uncertainties related to both the measurement of the pressure and the velocity. Following uncertainty can be expected according to the manufacture.

- Water level ± 0.3 mm
- Velocity $\pm$ 0.03 m/s

The sensor is set to conduct measurements in a temporal resolution of 1 minute. A detailed description of the area velocity sensor can be seen in the manufactures manual [Teledyne Isco, 2012].

5.3 Inspection of Case Data

Measurements conducted in the period October the 18th - November the 12th 2012 was used in order to test the developed methods. In figure 5.9 the measurements from the sensors are given.

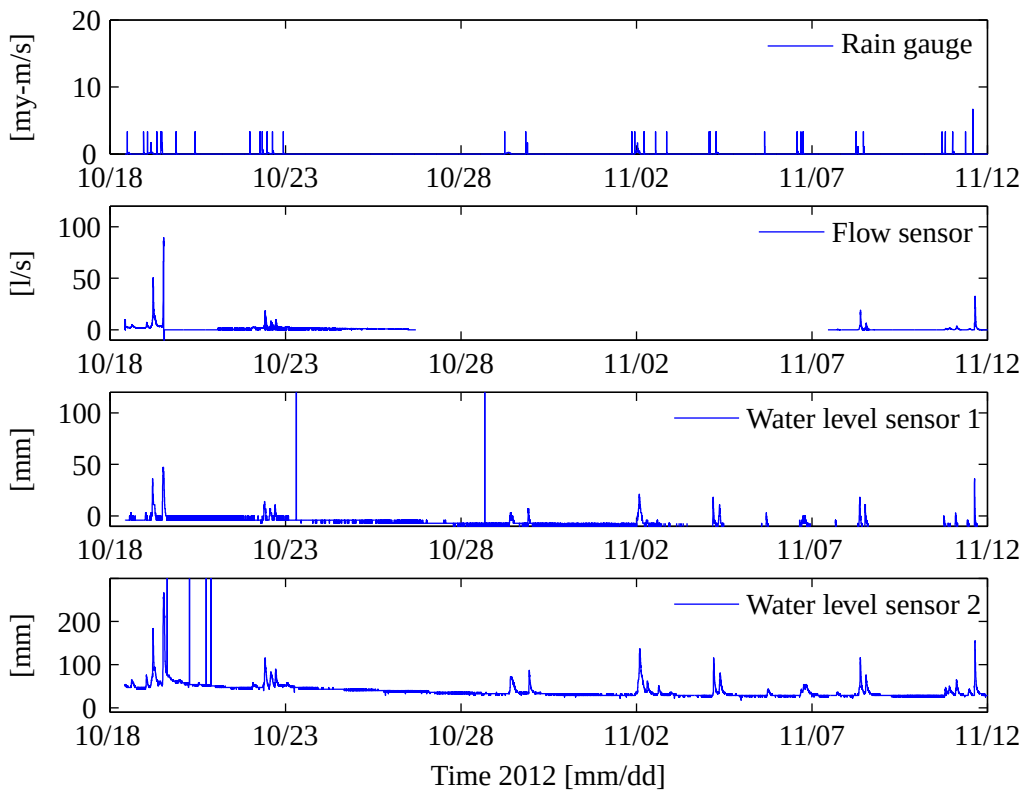


Figure 5.9. Data acquired in the period October the 18th - November the 12th 2012

The rain gauge given in figure 5.9 is located approx 2 km north west of the case catchment at Egaa WWTP. The rain gauge SVK 22321 is part of the Danish SVK rain gauge-system. [DMI, 2012b] The rain gauge measures the rain by a tipping bucket in a resolution of 0.2 mm.

In figure 5.9 the distance d measured by the water level sensors have been transformed to a water depth. For further explanation see figure 5.4.

By visual inspection of the graphs in figure 5.9 different kind errors can be detected.

The acquired data from the flow sensor were subject to different kind errors. More than 50 % of the data from the flow sensor was missing however the missing data was caused by lack of power and therefore this error could have been avoided.

If the missing data are neglected the main source of error is the velocity sensor, while the pressure transducer seems to conduct precise measurements. During dry weather conditions, where the flow can be assumed to be stationary and uniform, the maximum deviation in measurements from the pressure transducer correspond to 1 mm.

In the following some examples of the observed errors with respect to the velocity measurements are given. A negative flow rate was detected October the 19th see figure 5.10.

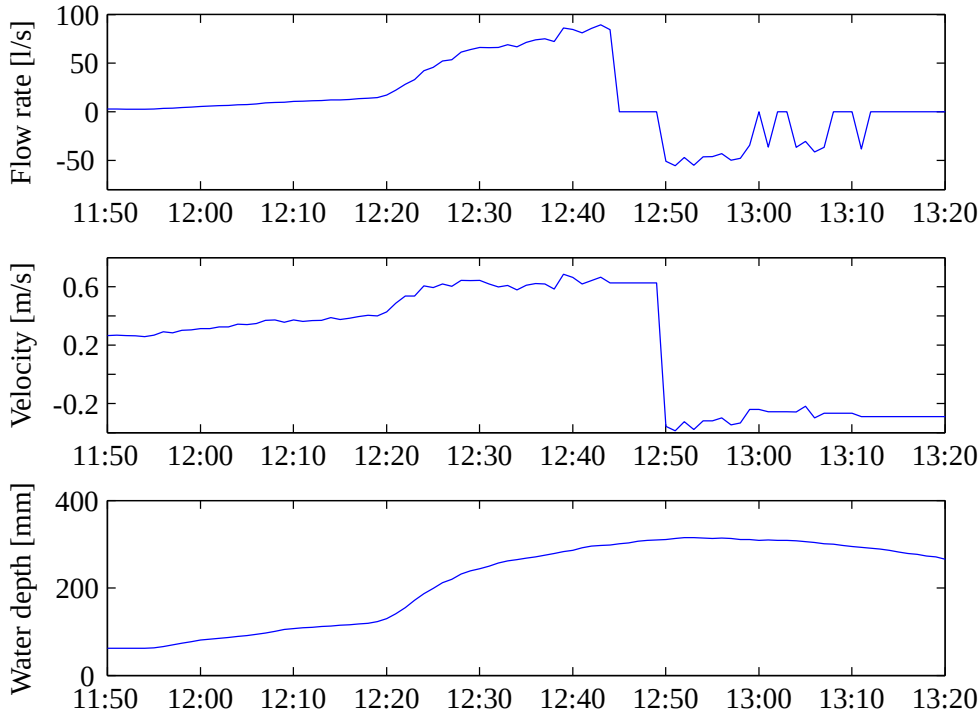


Figure 5.10. Flow, velocity and water depth measured the 19th Oktober 2012 in the period 11:50-13:20

Suddenly the measured velocity is negative, although the water depth is still increasing. The negative measured velocity results in a negative flow output from the sensor which obviously is not correct. The error will be identified by the initial control, since the measurement is a non physical value. The error is probably due to blocking of the ultra sonic signal.

In figure 5.10 it can also be seen that the flow rate sometimes is zero when the velocity is not zero.

This error can also be seen in another case see figure 5.11.

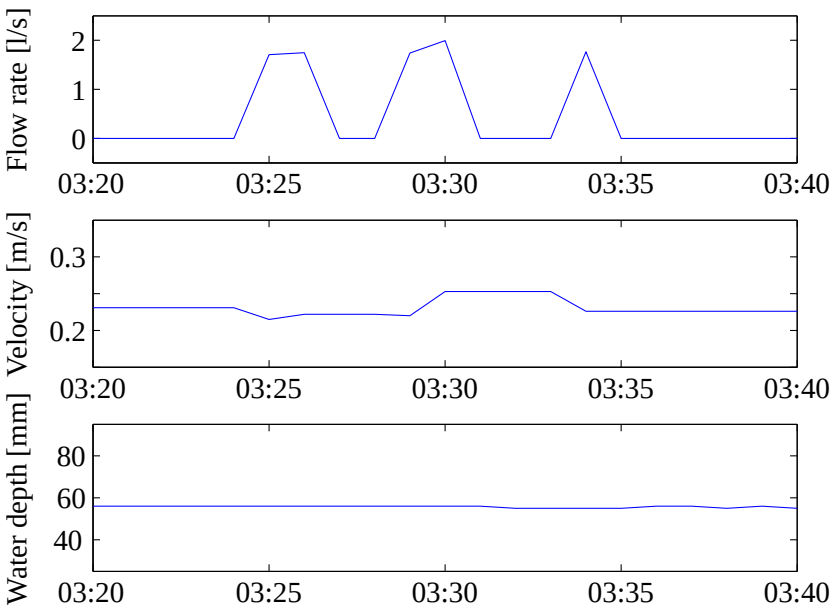


Figure 5.11. Water depth, velocity and flow the 21st Oktober 2012 in the period 03:20-03:40.

The event presented in figure 5.11 is a dry weather flow event.

When the velocity change the flow rate also change otherwise the flow rate is zero even though both a velocity and a water depth is measured.

The incoherent outputs from the sensor is due to preprogrammed configurations. If the sensor fails to measure the velocity at time *t*, it will set the velocity at time *t*, to the velocity measured at time *t-1*.

Although the sensor will set the flow to zero when no velocity is measured. Therefore a zero value of the flow can mean two things, either no flow is present or the sensor fails to measure the velocity. This error should be identified by the zero control. Since a dry weather flow is present at the flow sensor it should not be impossible to measure a velocity equal to zero.

With respect to the water level sensors different kind of errors have been detected. In some cases suddenly extreme values are present, see figure 5.12.

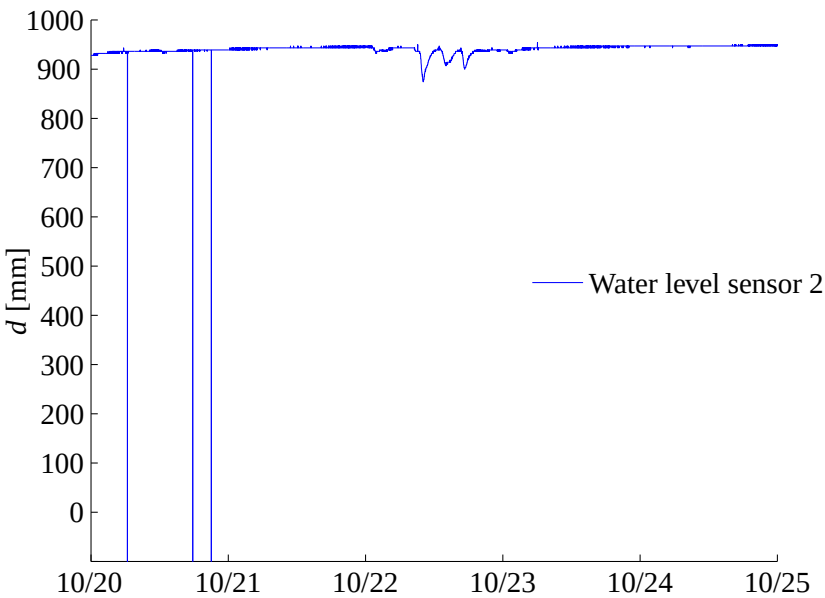


Figure 5.12. Sensor output from water level sensor 2. [mm/dd 2012]

As given in figure 5.12 suddenly measurements become negative, but it makes no sense to have a negative distance from the sensor head to the water surface, therefore this is obviously an error. When the negative output signal is transformed to a water depth it will result in water depth higher than d_0 - see figure 5.4. These errors should be detected by the initial control when testing for non physical values.

At water level sensor 1 the

opposite problem occurred, meaning that measurements larger than the distance from the sensor head to the bottom of the manhole was measured.
Again the error should be detected by the initial control when testing for non physical values.

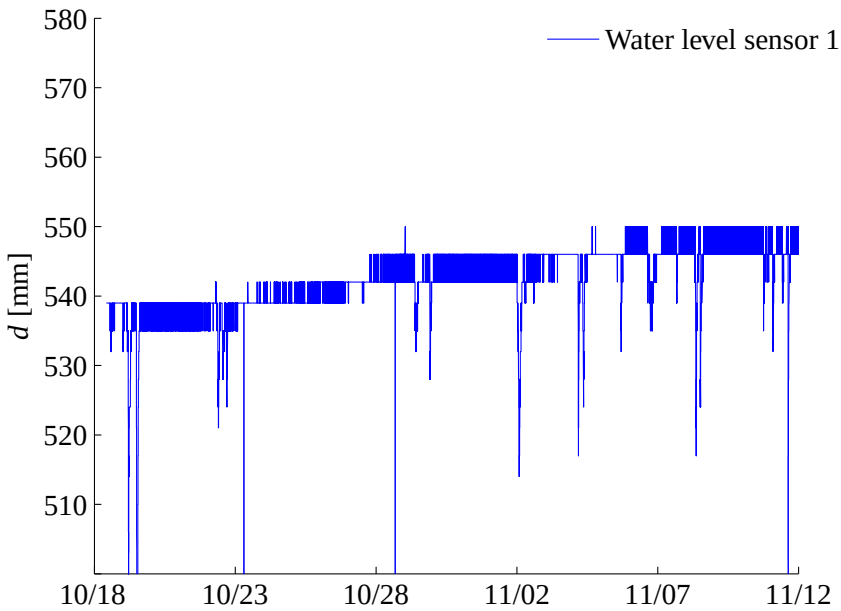


Figure 5.13. Sensor output at water level sensor 1

Furthermore is an increasing trend observed see figure 5.13. This indicate that the sensor need to be calibrated more frequently, or it should be taken into account that the zero point is changing.

The errors presented here are those which can be directly identified simply by visual studying the graphs. The inspection of the data is of importance since it can help setting parameters for the validation procedure.

5.4 System Configuration

The developed system have to be set up uniquely for the case study. The flow diagram of the validation procedure for the case study is given in figure 5.14.

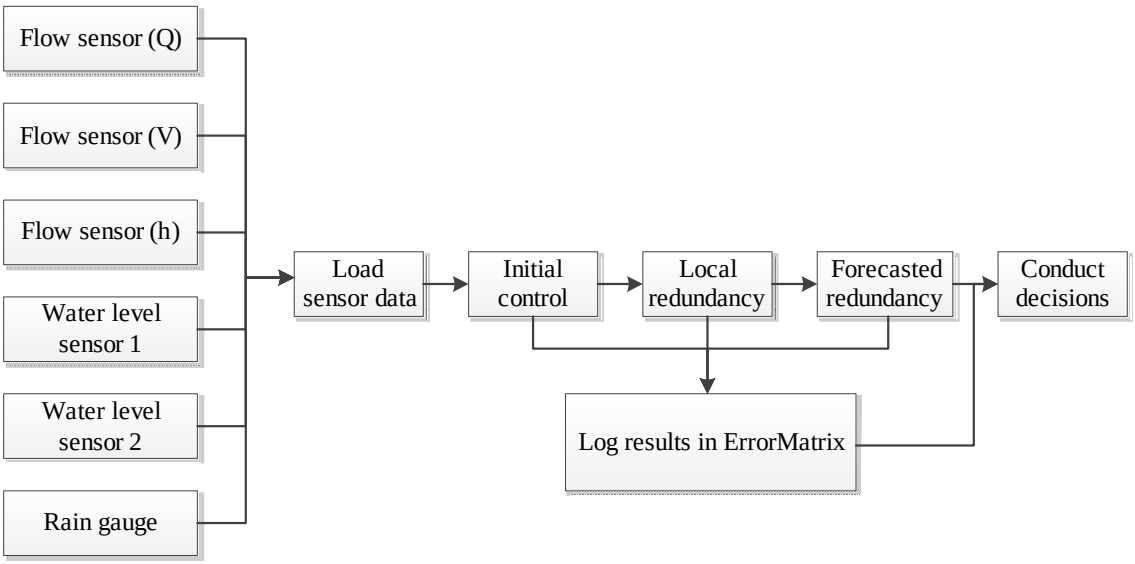


Figure 5.14. Validation flowchart for the case study

For the case study a total of six sensors are present. The flow sensor represents three sensors

(pressure, velocity and flow), there is two water level sensors and the rain gauge. The rain gauge will not be used directly in the validation procedure however it will still be loaded for visual inspection.

In order to conduct the steps given in figure 5.14 different parameters have to be set.

- Thresholds have to be set for the initial control in order to detect non physical values
- The acceptable percentage deviation have to be set for both the local and forecasted redundancy tests in order to detect abnormal residuals
- A Q-h-relation has to be established at the flow sensor based on the pressure transducer in order to conduct the local redundancy test.
- The flow routing model have to be calibrated in order to conduct the forecasted redundancy test

The assembled data will be divided into two parts. The first data period the 18th - 27th October will be used to calibrate parameters and the second data period the 7th - 11 Novemberth will be used to validate both the developed methods and the calibration. In the period 27th October - 7th November no data were assembled at the flow sensor due to lack of power as previously mentioned.

5.4.1 Parameters for Initial Control

The parameters to test for non physical values in the initial control have to be set for each sensor.

The lower threshold for all sensors are set to identify values less than zero, while the upper threshold is set unique for each sensor. The thresholds values can be seen in appendix B.1.

5.4.2 Acceptable Percentage Deviation for Redundancy Tests

The acceptable percentage deviation is set to 15 % in both redundancy tests. The setting of this variable determine how many errors are detected in the redundancy tests.

5.4.3 Calibration of Q-h-Relation at the Flow Sensor

Mannings equation is used as function to describe the Q-h-relations at all sensors.

Before the equation is fitted, obviously errors in the flow measurements are removed. The errors are identified by using the initial control to decide if data is valid. In figure 5.15 output from the flow sensor in the calibration period are given, before and after the initial control have removed identified errors.

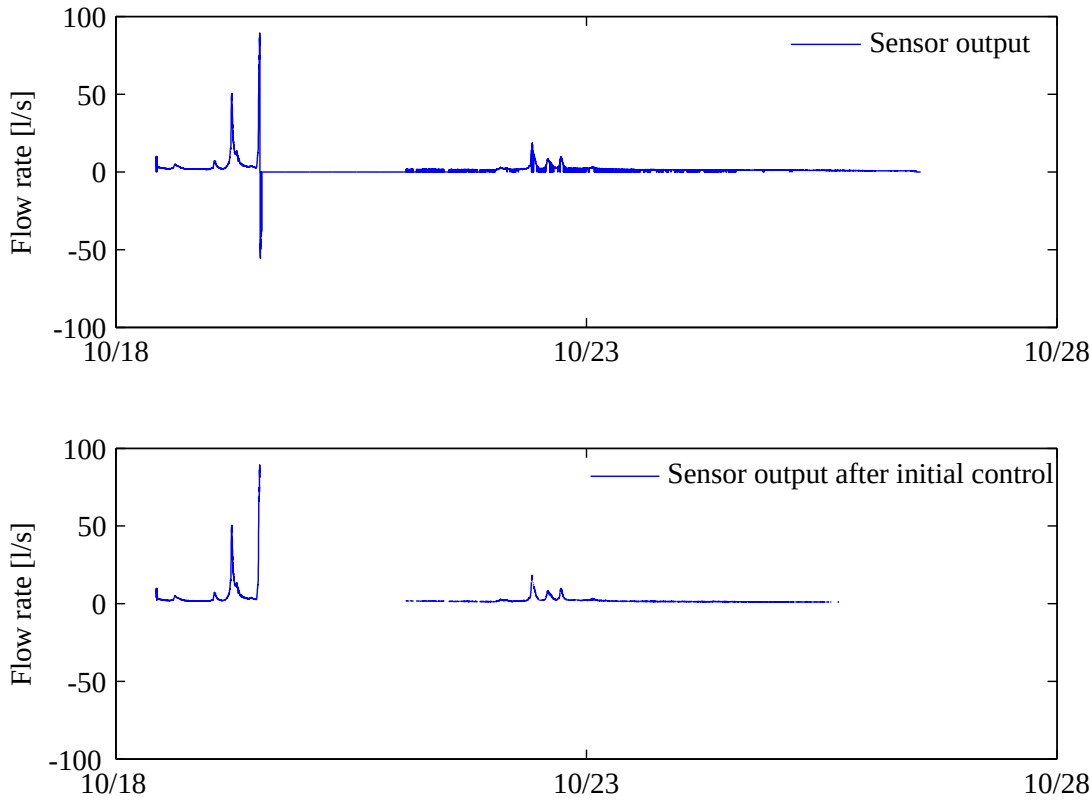


Figure 5.15. Flow data before and after the initial control have been used to remove errors

After the errors have been removed a total of 6139 data points are remaining. An additional correction is added to the pressure measurements. This is done since the flow sensor is compensating for the defined sedimentation depth of 30 mm when it is estimating the flow rate. The correction is applied by subtracting an area corresponding to a sedimentation depth of 30 mm from the calculated flow area when the flow rate is estimated based on the measured pressure.

In order to fit Mannings equation the least squares method is used to solve the overdetermined system [Ayyub and McCuen, 2003]. The sum of squared residuals are calculated according to equation 5.1

$$F = \sum_{k=1}^{K} (Qm_k - Qe_k)^2 \tag{5.1}$$

Qm	Measured flow rate	$[m^3/s]$
Qe	Estimated flow rate	$[m^3/s]$
K	Total number of data points (6139)	[-]

The estimated flow rate is calculated by Mannings equation as given in equation 5.2, under the assumption of stationary and uniform flow. Manning equations should be calibrated in order to estimate Qe so the sum of the squared residuals are minimised.

$$Qe_k = A(h_k) m R(h_k)^{2/3} \sqrt{I_0} \tag{5.2}$$

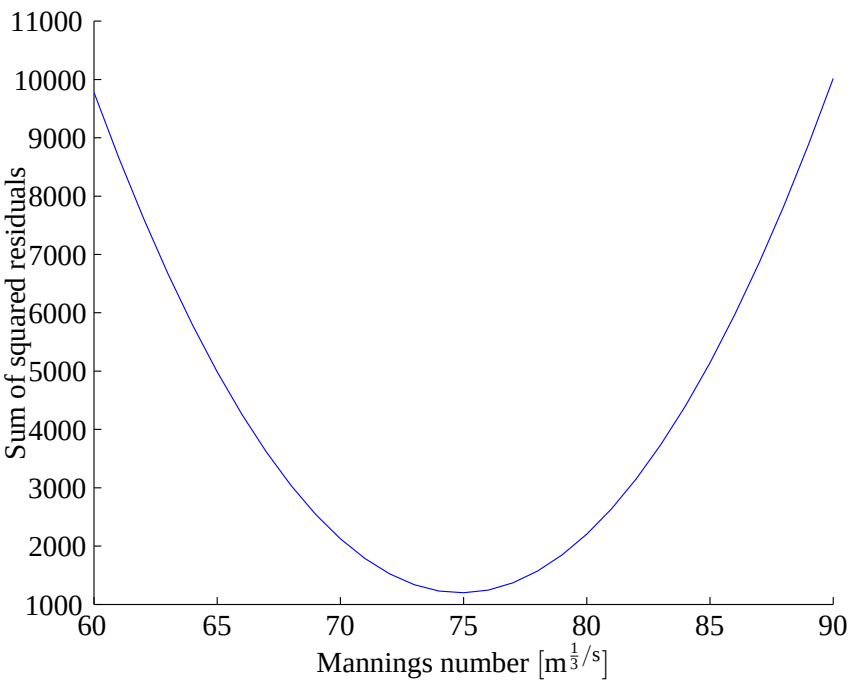


Figure 5.16. Result from least square optimisation

The target parameter in the calibration is chosen to be Mannings number evaluated in the interval 60-90 m$^{\frac{1}{3}}$/s. The sum of the squared residuals as a function of Mannings number is given in figure 5.16.

From figure 5.16 it can be seen that the minimum of the squared residuals are with Mannings number equal to 75 m$^{\frac{1}{3}}$/s.

In figure 5.17 the measured values are plotted together with the fitted model. Furthermore are the threshold to detect abnormal residuals given, based on the defined acceptable percentage deviation.

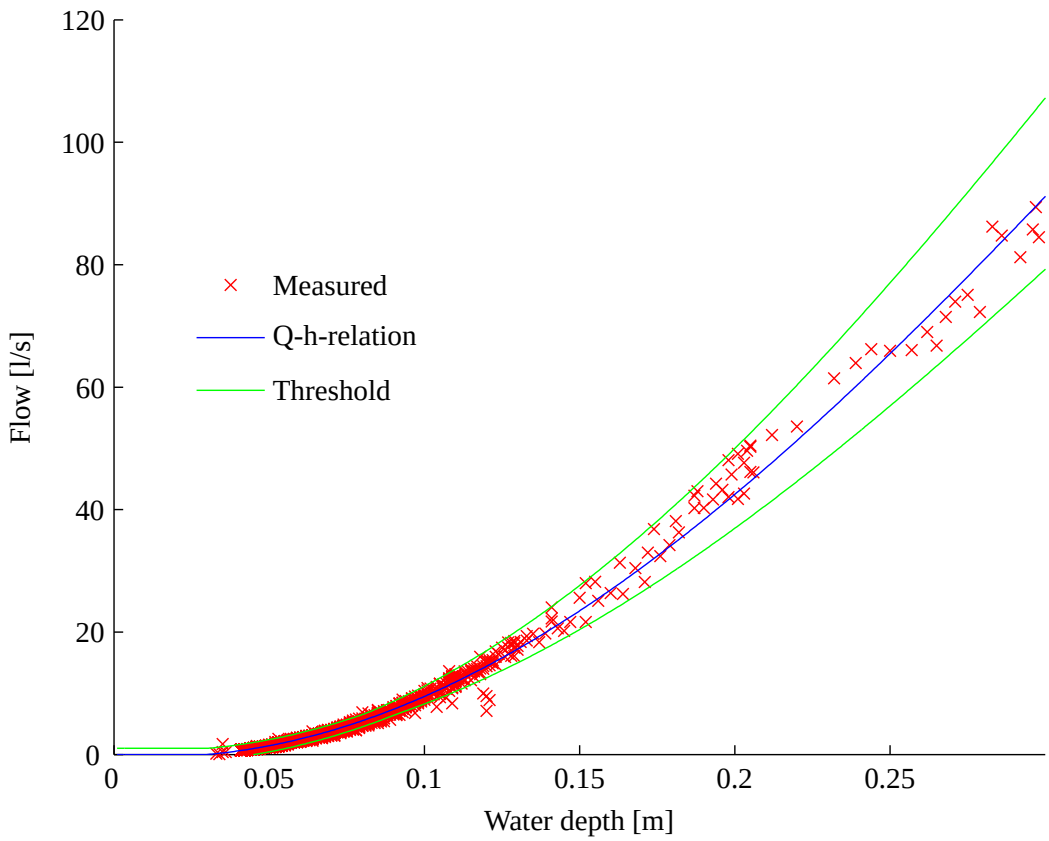


Figure 5.17. The fitted model plotted with measurements and threshold to detect abnormal residuals

The measurements within the thresholds are considered to be reliable measurements although corrupted by noise, while measurements outside are considered as errors.

5.5 Calibration of Flow Routing Model

The flow routing model is used to make the upstream water level sensors, redundant with the flow sensor. In chapter 4 the flow routing method was formulated as given in equation 5.3

$$Q_N(t) = \sum_{n=1}^{N-1} (Q_n(t - \tau_n^{'})) + \varepsilon \tag{5.3}$$

N	Total number of sensors	[-]
n	Index referring to individual sensors	[-]
$\tau_n^{'}$	Delay from sensor n to sensor N	[-]

Since the total number of sensors is three then N is three. Q_1 will be the flow rate from water level sensor 1, Q_2 will be the flow rate from water level sensor 2 and Q_3 will be the estimate of the flow rate at the flow sensor based on water level sensor 1 and 2. Mannings equation is used to estimate the velocity in all pipes, and thereby calculate the delays as described in chapter 4. This require that the Mannings number are calibrated in the manholes given in figure 5.18.

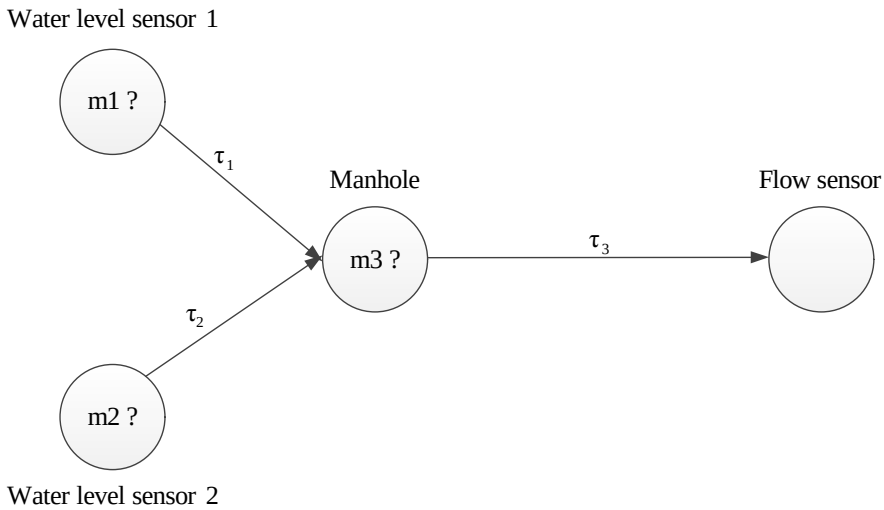


Figure 5.18. Simplified model of the case area

In the calibration the water level sensors are calibrated against the measurements obtained by the flow sensor. Again the same data set is used where errors have been removed by the initial control.

Each Manning number are evaluated in the interval 60-90 $m^{\frac{1}{3}/s}$. Again the least square method is used with all possible combinations of Mannings number corresponding to 29791 combinations. For each combination the delays are estimated in order to calculate Q_3 at the flow sensor.

The end result will be a 3-dimensional matrix where the optimal combination again will be the minimum value. The procedure is described in figure 5.19.

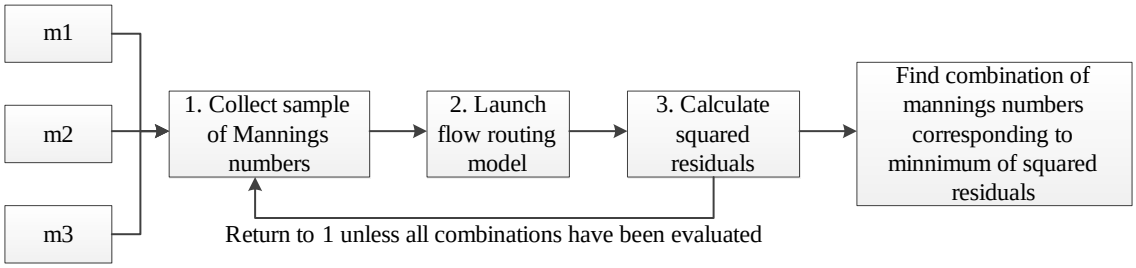


Figure 5.19. Flow of the calibration

The minimum of the squared residuals was obtained by the following Manning numbers.

	Mannings number [$m^{\frac{1}{3}}/s$]
m1	89
m2	73
m3	77

Table 5.3. Calibrated Manning numbers

In figure 5.20 the measured flow rates are plotted against the forecasted in the calibration period, with the calibrated Mannings numbers.

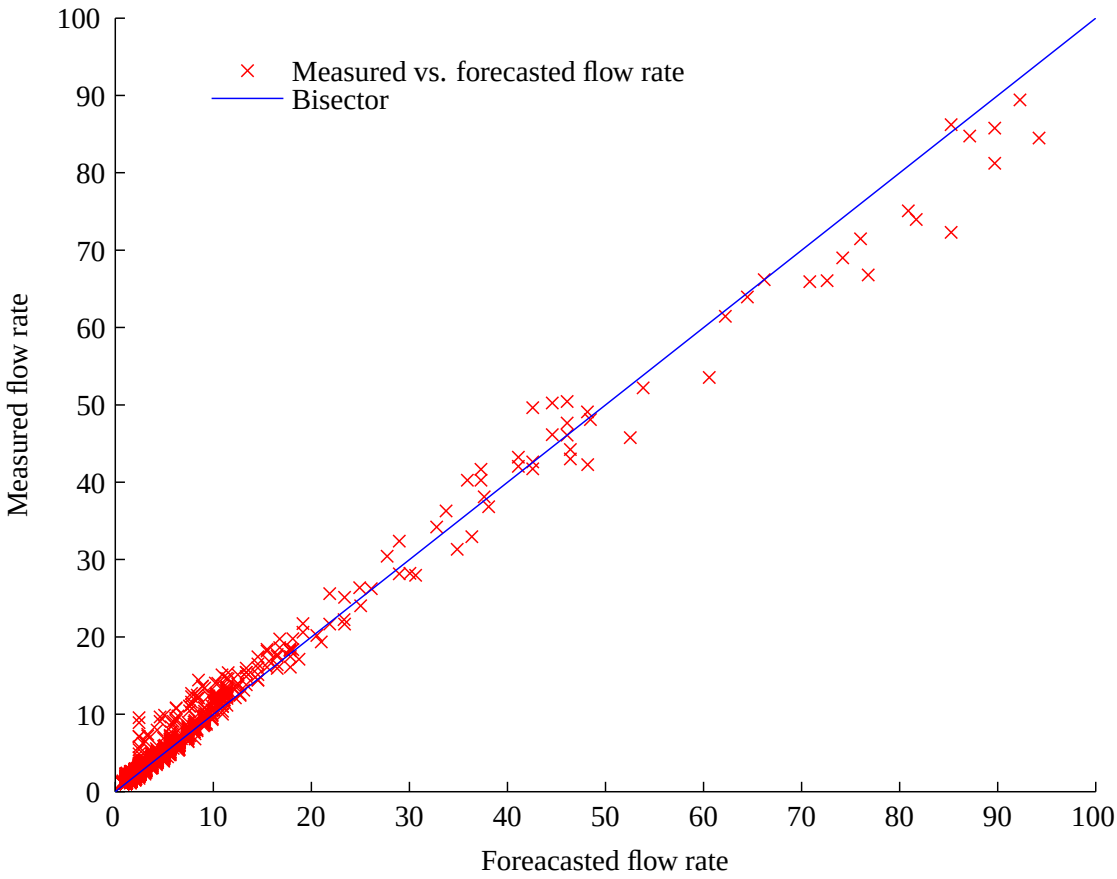


Figure 5.20. Measured against forecasted flow rates

5.6 Validation of Developed Methods

In the previous section the developed models was calibrated for the specific case area. The calibration seemed to produce satisfying results, although in order to validate the developed methods and the calibration, a validation is performed on another data set. The data which will be used for the validation is given in figure 5.21.

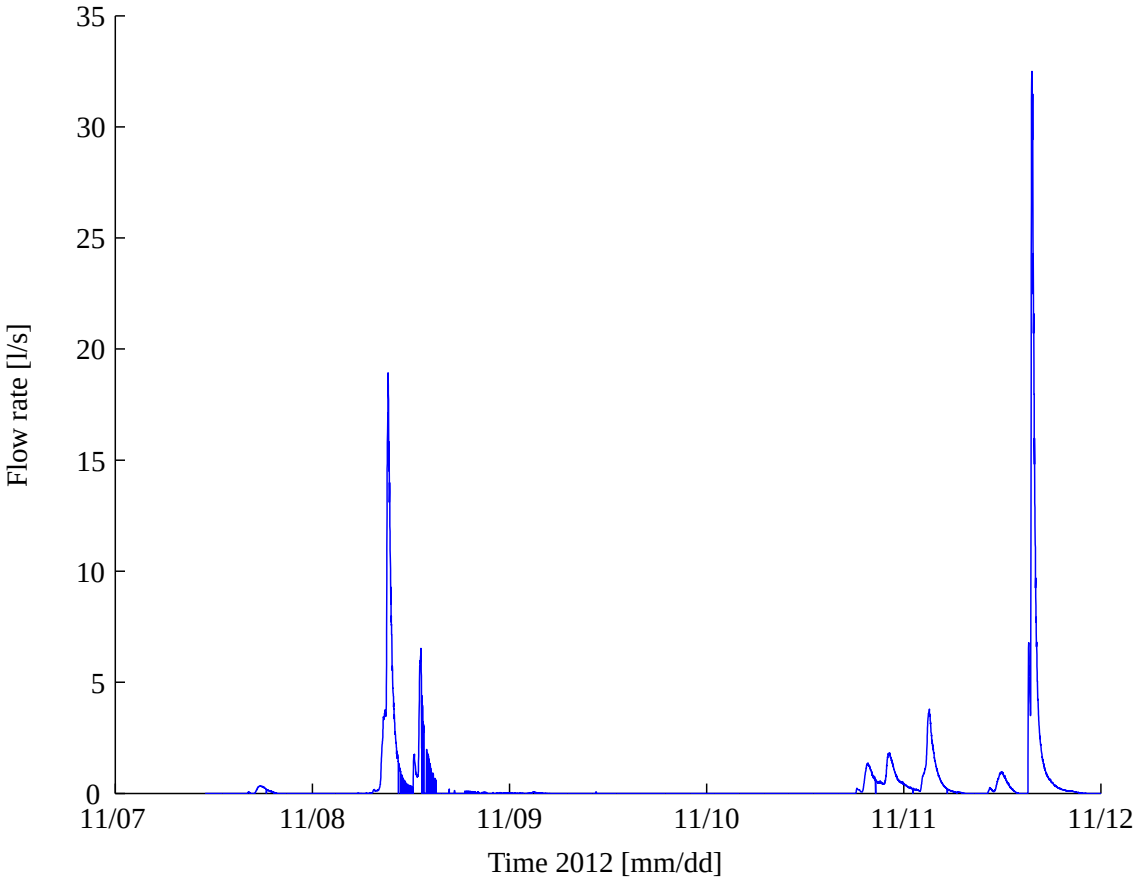


Figure 5.21. Flow sensor measurements in the validation period

The two events measured in the validation period will be analysed further.

The first event takes place the 8th of November 2012. For this event the output from the flow sensor is plotted in figure 5.22 together with the flow estimate based on water level sensor 1 and 2.

For the same event is the output from the flow sensor plotted in figure 5.23 together with the estimate based on the pressure transducer.

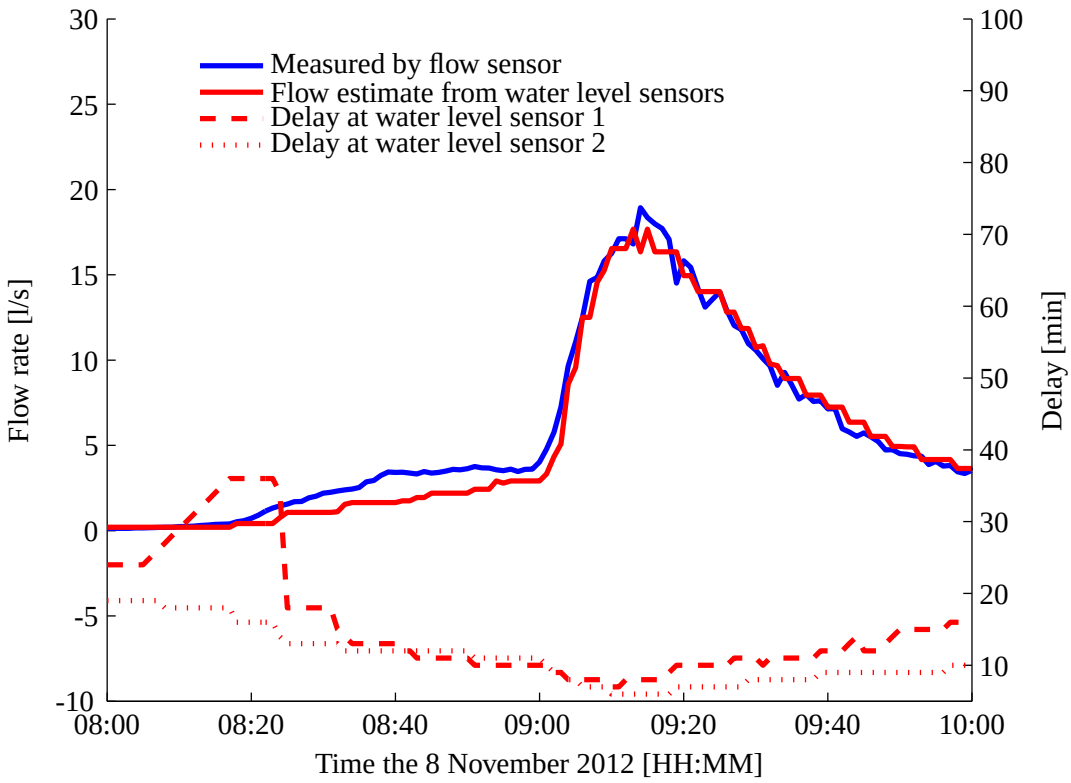


Figure 5.22. Measured and estimated flow rate

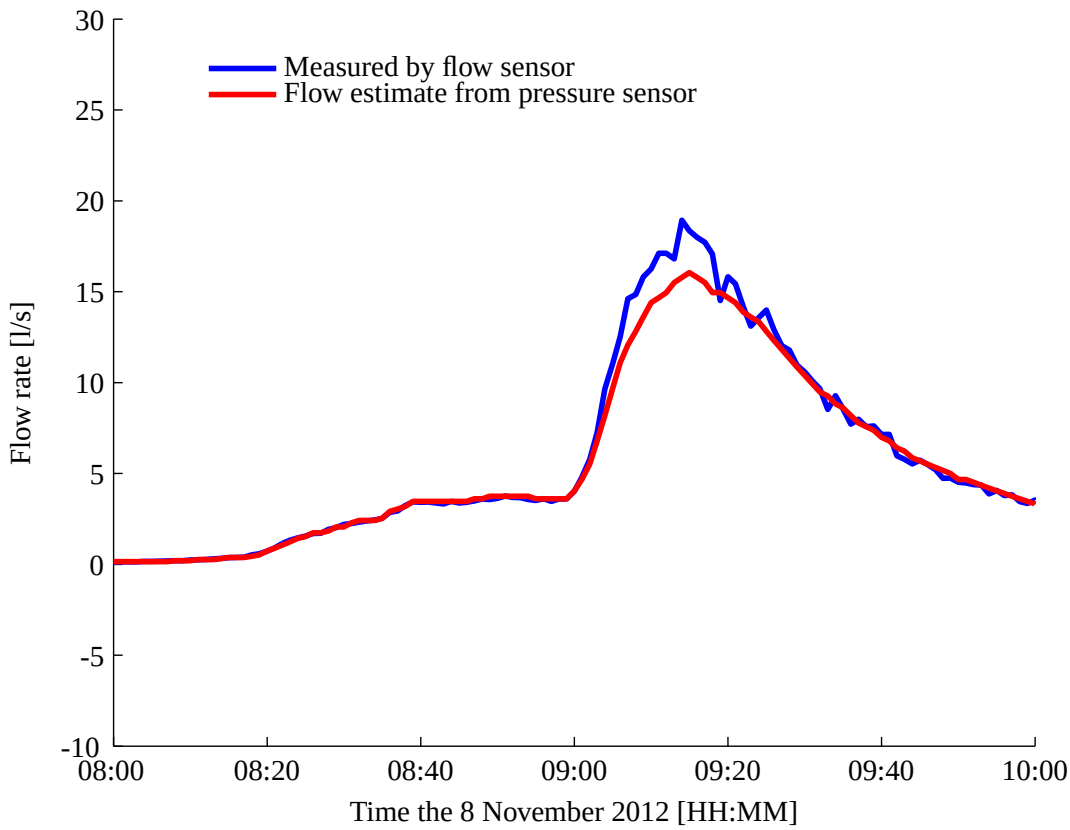


Figure 5.23. Measured and estimated flow rate

The estimate from the two water level sensors seems to fit well although in the start of the event the flow rate is underestimated, while the estimate from the pressure transducer underestimate the peak values.

The calculated residuals together with the calculated threshold based on the defined acceptable percentage deviation are given in figure 5.24.

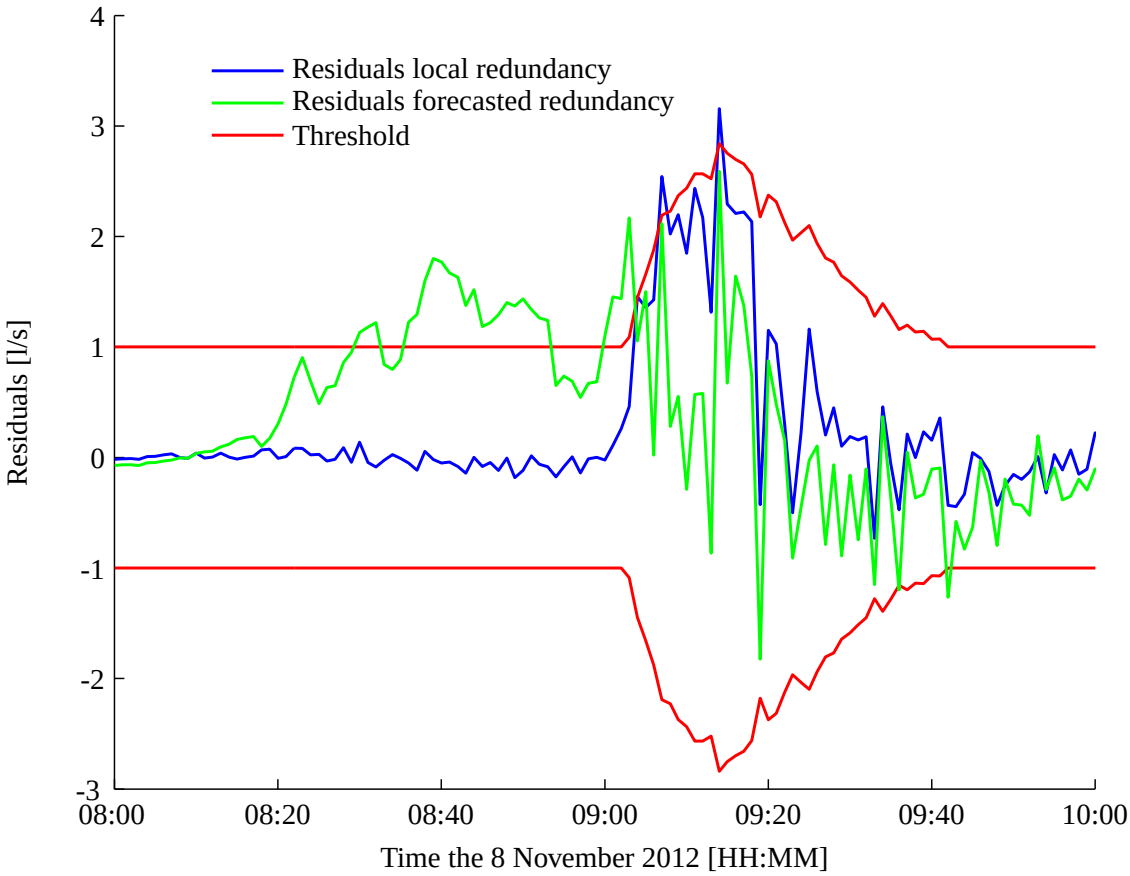


Figure 5.24. Calculated residuals over the event

The reason for the water level sensors to underestimate the start of the events, might be a result of surface run-off into the manhole between the water level sensors and the flow sensor.

The reason for the pressure transducer to underestimate peak values, might be due to the assumptions of stationary and uniform flow. Because the gradient of the energy level will be steeper at the wave front, and this is not included since the gradient of the energy level is set to the bottom slope. However the estimates from the water level sensors are also based on the assumption of stationary and uniform flow.

The trends observed for the event the 8th November 2012 can also be observed at the other event the 11th November 2012, see appendix B.2.

In order to investigate, how well the developed flow routing model perform compared with a full dynamic wave model the same event is modelled in the MIKE URBAN pipe network model. The setup is given in figure 5.25.

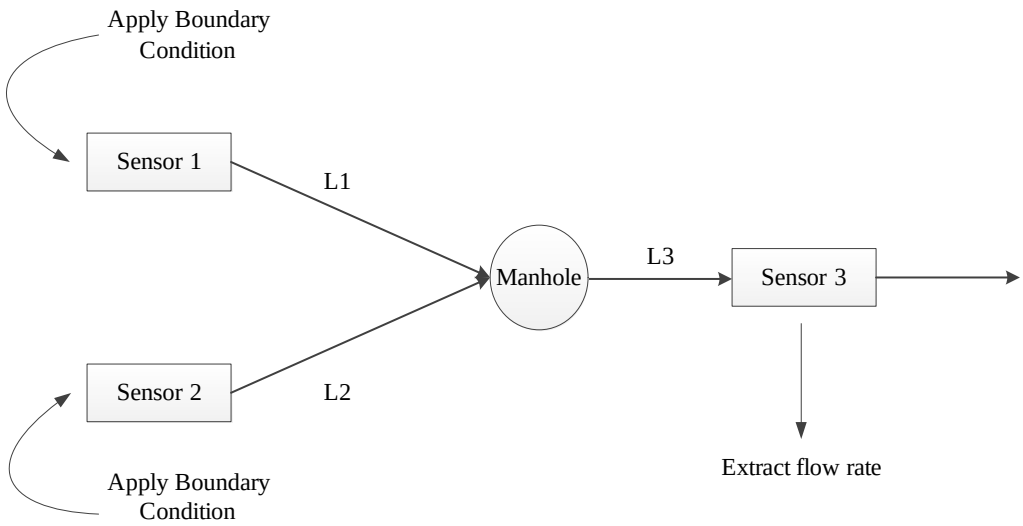


Figure 5.25. MIKE URBAN model setup

As boundary conditions, the measurements from the water level sensors are used after they have been transformed to a flow rate based on Mannings equation, with the calibrated Mannings numbers. In this way the comparison with the developed model, is only with respect to flow forecasting.

The Manning number for all pipes was set to 75 $m^{\frac{1}{3}}/s$. In figure 5.26 the MIKE URBAN result is given in comparison with the measured flow rate.

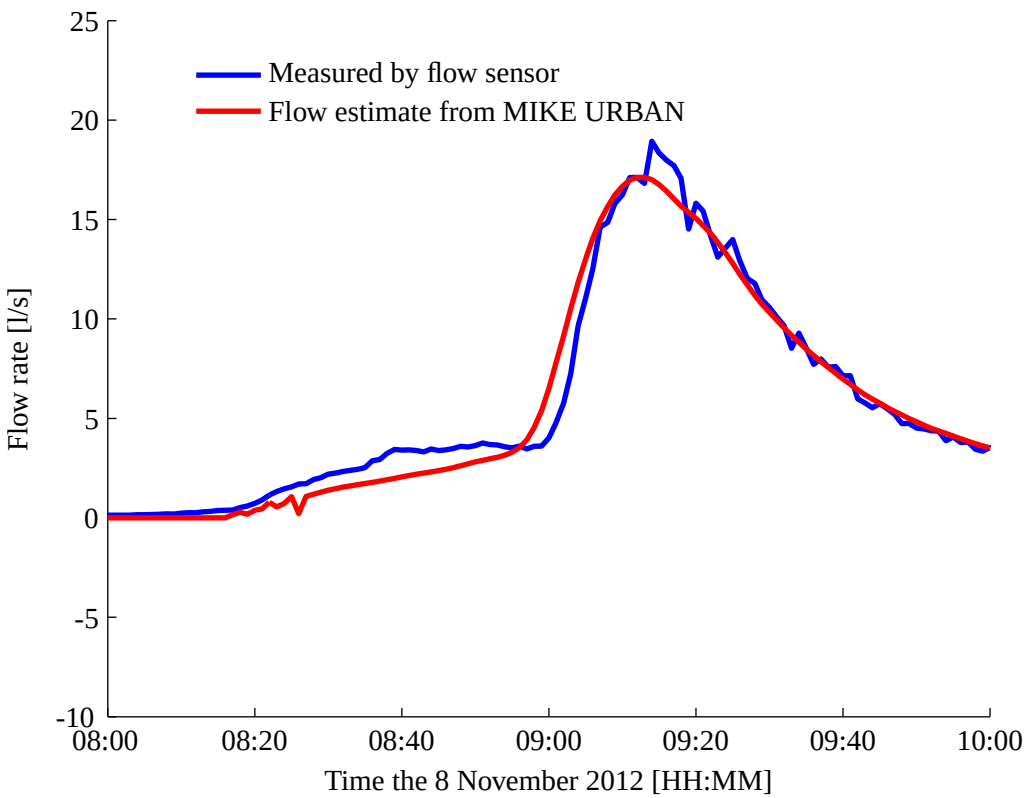


Figure 5.26. Comparison of measured flow and MIKE URBAN estimate

From figure 5.26 it does not seem like the MIKE URBAN pipe network model perform better than the developed flow routing model for the given event. Although no effort have been done in order to calibrate the MIKE URBAN pipe network model.

However it seems as the problem is diffusion of the wave and therefore the error is probably related to numerical dispersion. In order to minimise numerical errors the temporal and spatial discretisation was set to a minimum corresponding to respectively one second and one meter.

If the residuals for the two methods are compared it gives a better view of the performance. The residuals are defined as the modelled subtracted from the measured.

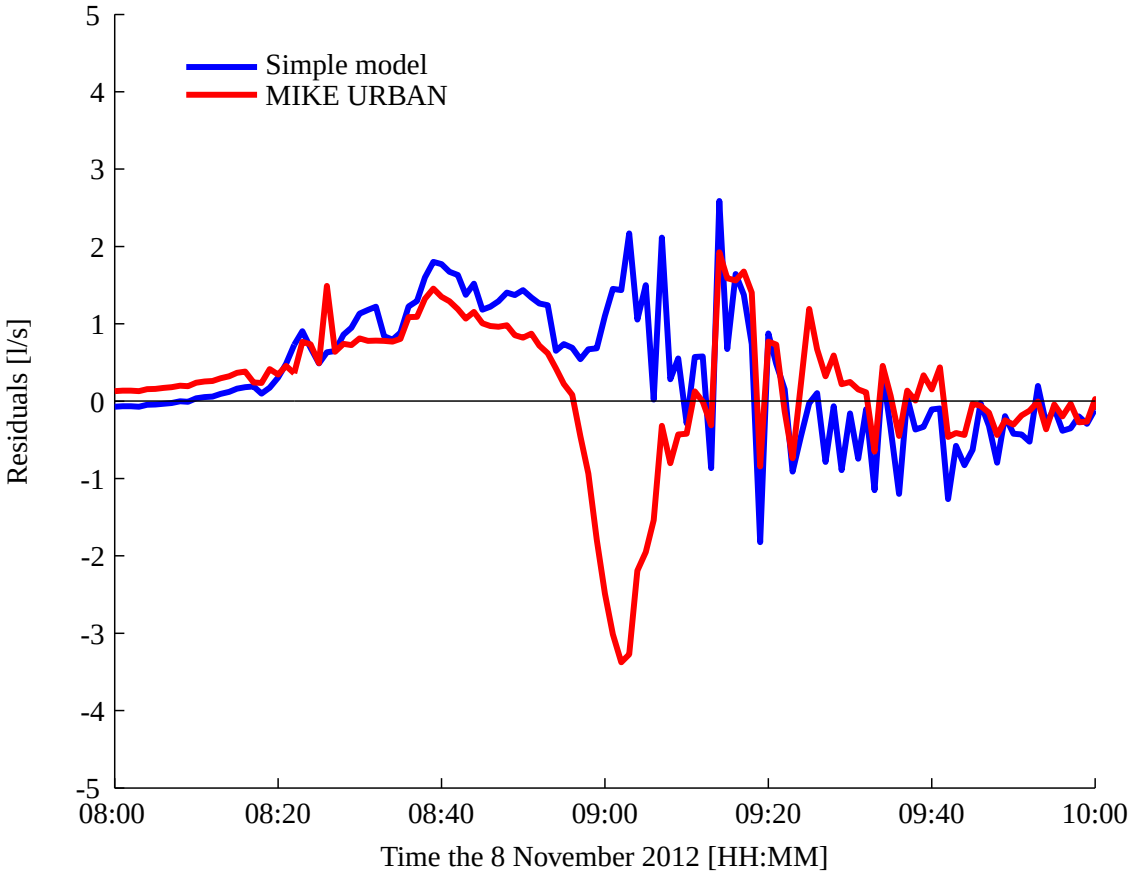


Figure 5.27. Simple model versus MIKE URBAN pipe network model

Figure 5.27 again indicate that the simple model perform well compared with the full dynamic wave model represented by the MIKE URBAN pipe network model.

From the events presented above it seems as the model to forecast the water level measurements produce satisfying results, but how does the validation procedure work. If the measured flow rate is plotted together with the output flow rate over the whole data period it will be as given in figure 5.28.

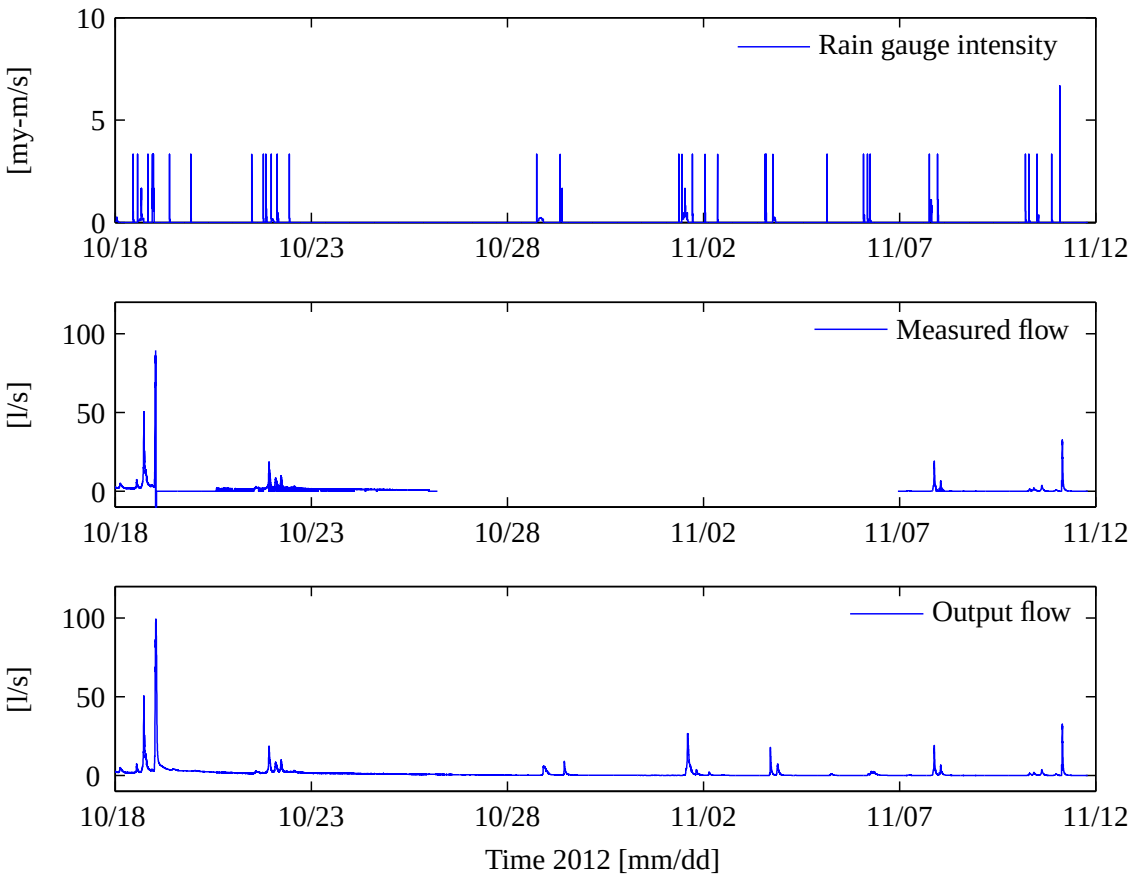


Figure 5.28. Comparison of measured flow rate and output flow rate

The time series is now complete since the missing data is substituted with an estimate based on the water level sensors, and furthermore are identified errors substituted, as an example see figure 5.29 which give a closer view on an event the 19th Oktober 2012.

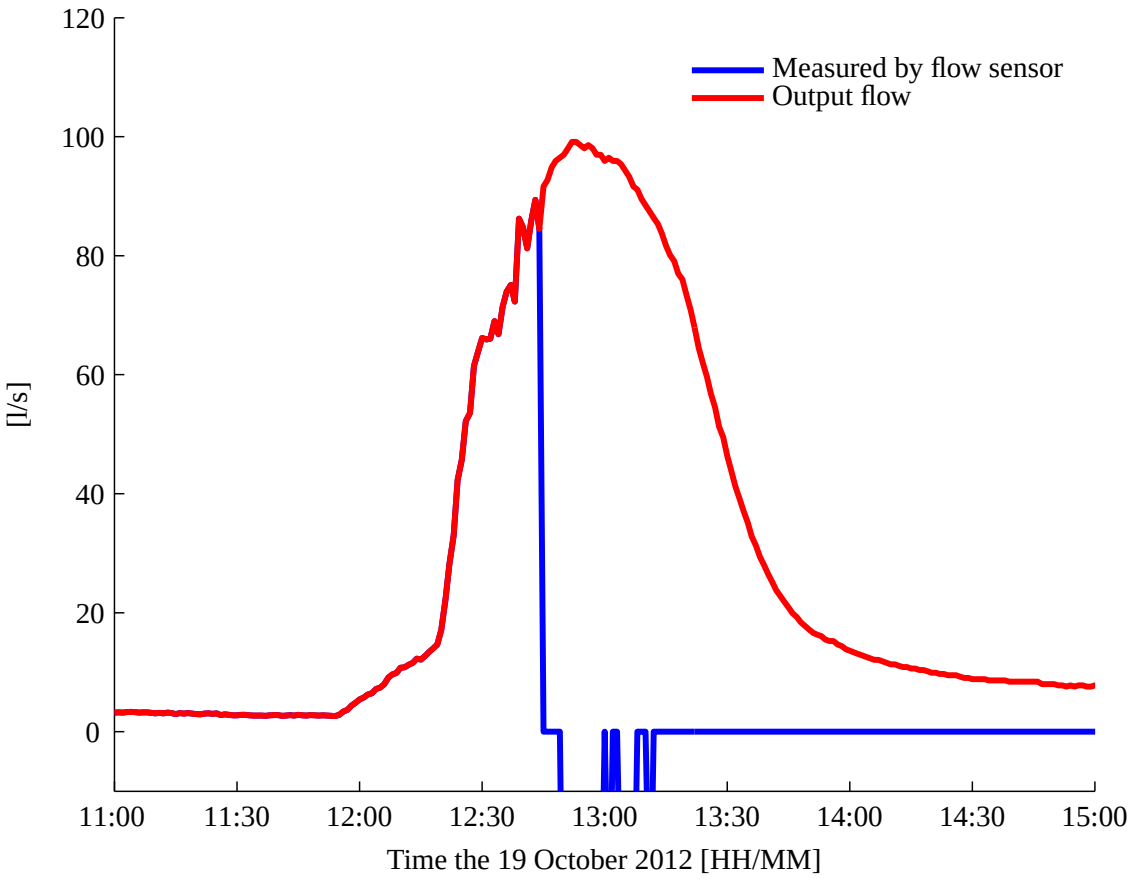


Figure 5.29. Comparison of measured flow rate and output flow rate

As seen in figure 5.29 detected errors are effectively substituted.

If the reader is interested in the raw MATLAB code used to analyse the case study it can be found in appendix D.2.

5.7 Summary

The case study proved that the simple flow routing model can produce reliable result. Furthermore sensor redundancy was used successfully to identify and replace outliers.

In the next part of the report, the possibility of combining different estimates in order to enhance the knowledge about the true state of a sewer system will be explored.

Part II

Data Assimilation

In the first part of this master thesis a technique for validating sensors based on redundancy were presented. When sensors are redundant then different estimates of the same variable is present. None of the different estimates correspond to the true value, but could the different estimates in combination give a better estimate of the truth than each estimate represent it self? This question was the motivation for the second part of this master thesis.

In 1960 Rudolf Kálmán's famous article "*A New Approach to Linear Filtering and Prediction Problems*" was published [Kálmán, 1960]. What he presented in this article is today referred to as the Kalman filter.

The methods which will be presented in the second part of the master thesis are all based on the Kalman filter. If the reader is unfamiliar with the Kalman filter the first chapter gives a gentle introduction to the subject.

The Kalman Filter 6

The Kalman filter consist of two main parts. The forecast part and the analysis part. In the forecast part the state variables are forecasted to next time step based on knowledge about the system which is measured.

The state variables are what is desired to get from the Kalman filter. It could for instance be water depth, flow rate etc.

It is not only the state variables which are forecasted, but also the errors associated to the forecast of the state variables. Furthermore is the Kalman filter described in a recursive form, meaning that only information from last time step is required in order to launch the algorithm at next time step. This feature make it adaptable for real-time applications.

In the analysis part the Kalman filter estimates the most probable estimate of the state variables, based on measurements and the model forecast. The Kalman filter also gives the errors associated to the new estimate.

The calculation cycle in the Kalman filter, is presented in figure 6.1

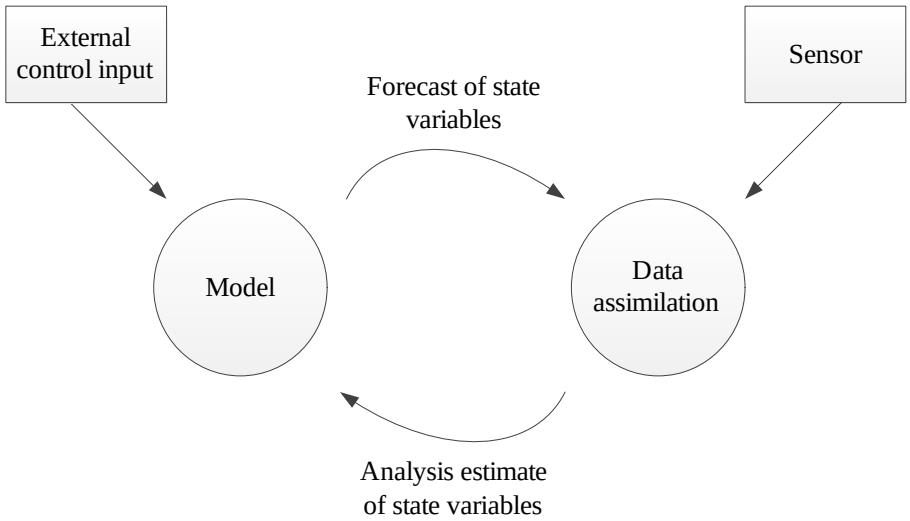


Figure 6.1. The calculation cycle of the Kalman filter

The external control input to the model can for instance be a boundary condition.

The general theory behind the analysis part of the Kalman filter can be explained by reviewing a simple example with one state variable.

Assume that a forecast of the state variable is present based on system knowledge, and the state variable is also measured. Furthermore assume that the errors related to both the forecast and the measurement of the state variable, are Gaussian with a zero mean (μ) and standard deviation (σ),

then the most probable estimate of the true value can be obtained by equation 6.1.

$$x^a = \frac{x^f \sigma^{2,z} + z\sigma^{2,f}}{\sigma^{2,f} + \sigma^{2,z}} \tag{6.1}$$

x^a	Analysis estimate of state variable	[-]
x^f	Forecast of state variable	[-]
z	Measurement of state variable	[-]
$\sigma^{2,f}$	Variance of error on forecast	[-]
$\sigma^{2,z}$	Variance of error on measurement	[-]

Equation 6.1 can also be used to assimilate two measurements directly.

The variance of the error on the analysis estimate can be calculated by equation 6.2.

$$\sigma^{2,a} = \frac{\sigma^{2,f}\sigma^{2,z}}{\sigma^{2,f} + \sigma^{2,z}} \tag{6.2}$$

[Rojas, 2003]

The relationship between the true unknown value (x) of the state variable and the analysis estimate in equation 6.1 is given in equation 6.3.

$$x \sim N(x^a, \sigma^{2,a}) \tag{6.3}$$

In figure 6.2 the principle is visualised, where it is assumed that the state variable is the velocity in a single point.

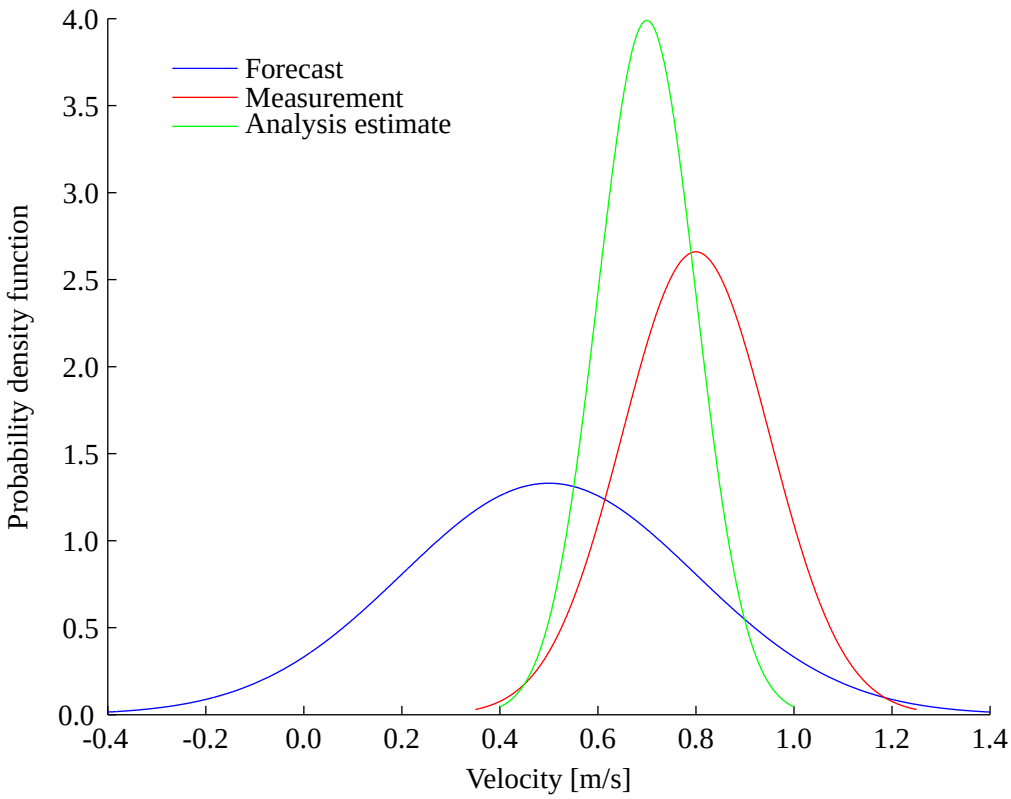


Figure 6.2. Visualisation of the analysis part in the Kalman filter, where the state variable and measurement both are the velocity in a single point.

The Kalman filter is based on the same assumptions as given above. It also calculate the most probable estimate, between a forecast of the state variables and a measurement under the same assumptions. However the Kalman filter operates with a variable called the Kalman gain (K). The Kalman gain can be derived by rewriting equation 6.1.

$$x^a = x^f + K(z - x^f) \tag{6.4}$$

Where the Kalman gain is expressed as.

$$K = \frac{\sigma^{2,f}}{\sigma^{2,f} + \sigma^{2,z}} \tag{6.5}$$

[Rojas, 2003]

The difference between the forecast and the measurement in equation 6.4, is often referred to as the innovation, and the innovation multiplied with the Kalman gain correspond to the update of the state variable.

For the Kalman filter to update the error associated to both the forecast and the analysis the system model must be linear as given in the equations below.

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1} \\ z_k &= Hx_k + v_k \end{aligned} \tag{6.6}$$

k	Time index	[-]
x	State variables, vector with size N and index n	[-]
A	Transition matrix with size N x N	[-]
u	Control input, vector with size J and index J	[-]
B	Transition matrix with size N x J	[-]
w	Process noise. Vector with size N	[-]
z	Measurements vector with size M, and index m where $M \leq N$	[-]
H	Transition matrix with size M x N	[-]
v	Measurement noise. Vector with size M	[-]

[Welch and Bishop, 2006]

The first equation describe how the state variables at last time step are forecasted to the state variables at next time step. This equation is often referred to as the state equation. This equation describe the physics of the system which is measured. The matrix A define how the analysis estimate of the state variables will be propagated to next time step. The matrix B is used to model how some external input control u influence the state variables. The model describing it is corrupted by the process noise w which is assumed to be Gaussian distributed with zero mean.

The second equation describes how the measurements z are linked to the state variables. This equation is often referred to as the output equation. The dimension of z can be smaller than x, since all wanted state variables not necessarily need to be measured. The matrix H describe how the measurements are linked to the state variables. The measurement is corrupted by the measurement noise v, again assumed to be Gaussian distributed with zero mean. v is a vector with same size as the vector z.

When the system model is defined the filter is basically ready for operation. The calculation steps in the algorithm can be seen in figure 6.3 and in table 6.1 are the definitions of the parameters given.

The calculation of the Kalman gain as presented in figure 6.3, are different to what presented in equation 6.5. This is because the Kalman filter is capable of operating with multiple state variables and measurements. Furthermore the Kalman filter operates with covariances.

In order to explain the algorithm, each step is described, although for a complete derivation of the Kalman filter algorithm, the reader should see [Ribeiro, 2004] or [Kálmán, 1960].

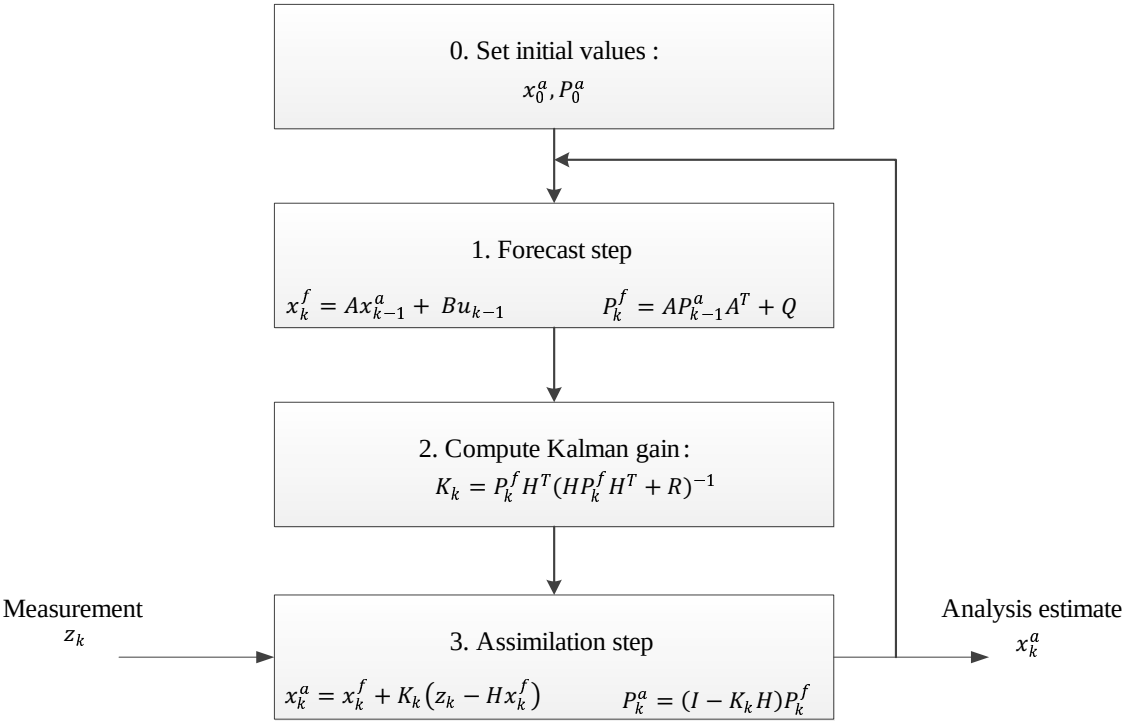


Figure 6.3. The Kalman filter algorithm, inspired by [Rojas, 2003], [Kim, 2010] and [Ribeiro, 2004]

Variable	Description	Variable	Description
x^a	Analysis estimate of state variables	A^T	Transposed of A
P^a	Covariance of error on analysis estimate	Q	Covariance of process noise
x^f	Forecast estimate of state variables	K	Kalman gain (weighting factor)
P^f	Covariance of error on forecast estimate	H	Matrix from system model
A	Matrix from system model	H^T	Transposed of H
B	Matrix from system model	R	Covariance of measurement noise
I	Identity matrix	k	Time index

Table 6.1. Definition of variables in the Kalman filter

0. Set Initial Values

The initial values x_0^a and P_0^a have to be set in order to launch the calculation at time $k = 1$.

P_0^a is the covariance matrix of the error on the analysis estimate which have the size n x n. The definition of P^a is given in equation 6.7.

$$P^a = E[(x^a - x)(x^a - x)^T] \tag{6.7}$$

x	True but unknown state variables	[-]

The error covariance describe how the errors on the state variables are correlated. The error covariance make it possible to update state variables which not are measured.

With knowledge about the initial state of the system, this should be used to set the initial conditions. However over time the setting of the initial conditions will have a minor impact on the estimate.

1. Forecast Step

In this step a forecast of the state variables is calculated based on system knowledge. This is done by equation 6.8 which is given by the system model.

$$x_k^f = Ax_{k-1}^a + Bu_{k-1} \tag{6.8}$$

The process noise is used to propagate the error covariance matrix (P), so it corresponds to the errors associated to the new forecast of the state variables. This is what happens in equation 6.9.

$$P_k^f = AP_{k-1}^a A^T + Q \tag{6.9}$$

The matrix Q is the covariance of the process noise w, which have the dimension n x n. Q is given by equation 6.10.

$$Q = E(ww^T) \tag{6.10}$$

The setting of Q is of importance for the weighting between the forecast and the measurement.

II. Compute Kalman Gain

With a forecast of the state variables and the corresponding covariance, the algorithm is ready to calculate the Kalman gain. The Kalman gain is calculated according to equation 6.11.

$$K_k = P_k^f H^T (HP_k^f H^T + R)^{-1} \tag{6.11}$$

A large Kalman gain will result in a higher weighting of the measurements, therefore if the noise on the measurements are low the Kalman gain increase. R is calculated by equation 6.12.

$$R = E(vv^T) \tag{6.12}$$

III. Assimilation Step

The Kalman gain is used to calculate a new estimate of the state variables. This is done by equation 6.13.

$$x_k^a = x_k^f + K_k(z_k - Hx_k^f) \tag{6.13}$$

As in the simple example with one state variable, the Kalman gain determines how much of the innovation, that should be added to the forecasted state variables.

Finally the error covariance of the estimate is updated so it corresponds to the analysis estimate of the state variables. This is done based on the calculated Kalman gain according to equation 6.14

$$P_k^a = (I - K_k H)P_k^f \tag{6.14}$$

6.1 Summary

In this chapter the linear Kalman filter was presented. In the next chapter it will be applied in order to remove measurement noise. In some cases it is not possible to make a linear system model, but in this case other versions of the Kalman filter can be used. In chapter 8 a version called the Ensemble Kalman filter will be used.

The Kalman Filter for Assimilation of In-situ Sensors 7

Redundant sensors can directly be assimilated by equation 6.1 on page 60, if the noise on both sensors are Gaussian with zero mean. This could for instance be done in a retention basin, where both a pressure sensor and a water level sensor are installed. In this case it is only the assimilation part of the Kalman filter which is necessary.

If the redundant sensors are a velocity sensor and a pressure sensor, they can also be assimilated for instance to improve the velocity estimate.

The pressure measurement could be used to model a control input to the Kalman filter. By establishing a V-h-relation, and then evaluating how the velocity estimated by the V-h-relation, is changed between time steps, then the control input in form of an acceleration of the water can be calculated. If the flow is assumed to be quasi stationary and uniform, and Mannings equation is used to describe the V-h-relation, then the acceleration can be calculated according to equation 7.1.

$$u_{k-1} = \frac{V_k - V_{k-1}}{\Delta t} = \frac{mR(h_k)^{2/3}\sqrt{I_0} - mR(h_{k-1})^{2/3}\sqrt{I_0}}{\Delta t} \tag{7.1}$$

With the pressure measurement transformed to a control input in form of an acceleration then the system model can be described as a linear system, see the equations below. The state variable x and the measurement z are both set to be the velocity. Thereby the size of both x and z are 1 x 1.

$$\begin{aligned} x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \Leftrightarrow x_k &= x_{k-1} + \Delta t u_{k-1} + w_{k-1} \\ z_k = Hx_k + v_k \Leftrightarrow z_k &= x_k + v_k \end{aligned} \tag{7.2}$$

According to the system model, the matrices A, B and H, are constant in time, and due to the the size of x and z, all the matrices will only have one entrance and can therefore be considered as scalars. Both A, H are set to one and B is set to equal to Δt. Thereby the first equation simply state that the velocity at time k is equal to the velocity at time $k-1$ plus the acceleration multiplied with the time step, and then added with some random process noise. The second Equation state that the measurement of the velocity is equal to the velocity added with some random measurement noise.

The measurement noise v would have to be estimated if it is not provided by the manufacturer of the flow sensor. To derive the exact value for the process noise is not possible with the given system model. It cannot be analytical determined how the error on the pressure measurements will be reflected in the estimated acceleration. Furthermore the error on the estimate is not constant. If the flow conditions are changing rapidly the error on the estimate increase due to the assumption of stationary and uniform flow. The process noise is considered as a calibration factor.

7.1 Test on Data From the Case Study

In chapter 5 the Manning number was calibrated to 75 $m^{\frac{1}{3}/s}$ at the flow sensor. Based on the calibrated Manning number a V-h-relation is established. The V-h-relation together with the measured velocities are given in figure 7.1.

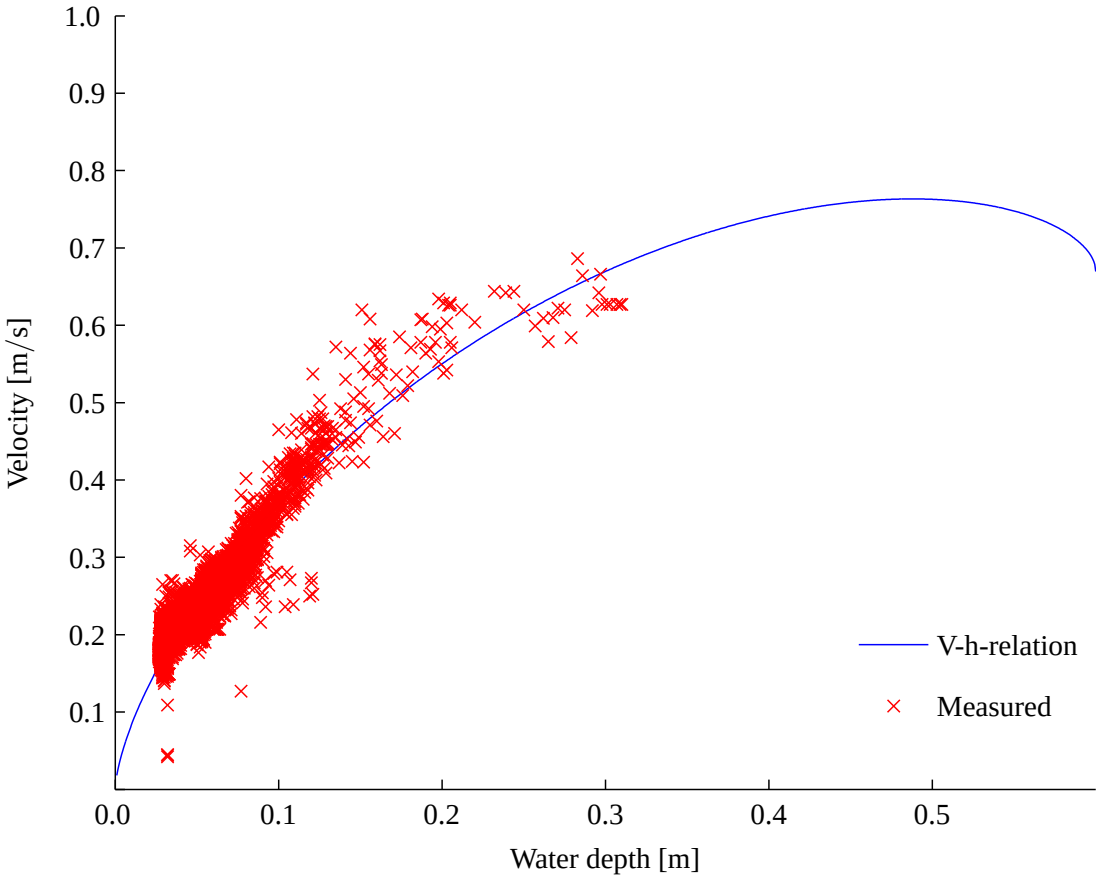


Figure 7.1. V-h-relation with measured velocities

For the Kalman filter to be theoretical correct the measurement noise needs to follow a normal distribution. To investigate whether this is the case and to estimate the standard deviation of the measurement noise, a period with steady flow is inspected see figure 7.2.

In a five hour period the water depth was measured to a constant value while there were fluctuations on the velocity measurement. These fluctuations are therefore assumed to be a result of noise. By plotting the measurements in a histogram it can be seen that the noise can be assumed to follow a normal distribution see figure 7.3.

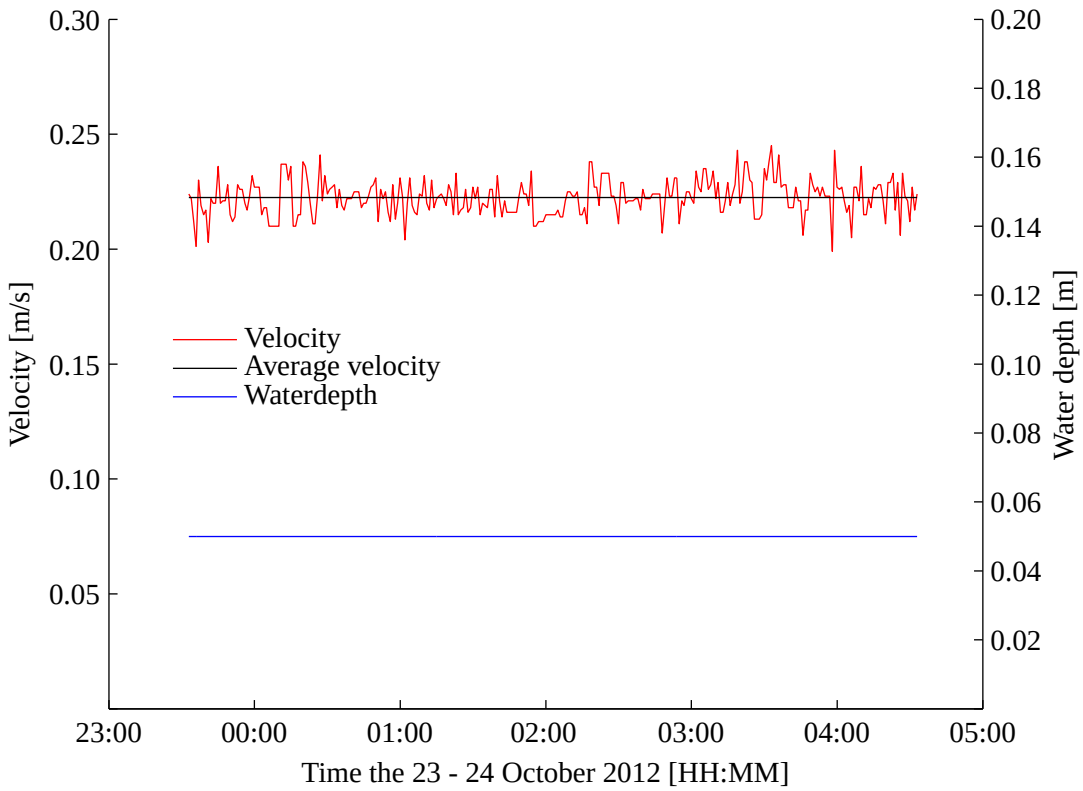


Figure 7.2. Water depth and velocity measurements in a five hour period

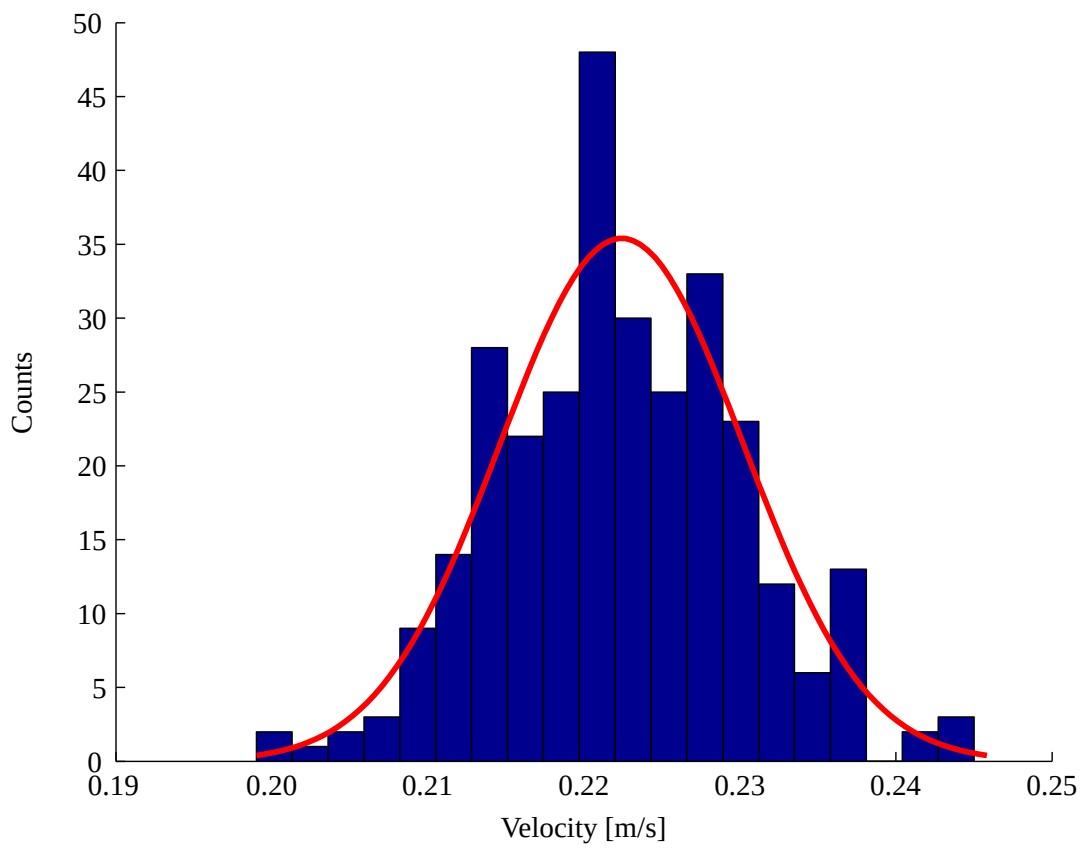


Figure 7.3. Histogram of velocity measurements given in figure 7.2

The standard deviation is calculated to 0.008 [m/s] which will correspond to the measurement noise v. The measurement noise covariance R will be the the squared standard deviation of the measurement noise corresponding to $(0.008\ \mathrm{m/s})^2$. R is constant in time.

As mentioned the process noise w can not be estimated analytical, and is therefore considered a calibration factor.

In situations where the flow is close to be stationary and uniform the acceleration estimated by the pressure is precise, because the only error source will be from the noise on the pressure sensor. However in case of waves in the sewer system the precision is low since the gradient of the energy level not correspond to the bottom slope. Therefore the process noise is not set to be constant.

If the change in pressure between measurements continuously are calculated, then this information can be used to set the process noise. The idea is to rely on the velocity measurements when the the pressure is changing rapidly, because in this case we are aware that the error on the forecast from the system model increases. On the other hand, in periods with constant pressure more reliability is given to the forecast from the system model.

The process noise covariance (Q) is set to vary linear based on the change in pressure between time steps ($\frac{\Delta p}{\Delta t}$) as given in table 7.1.

	Min	Max
Q $[(\mathrm{m/s})^2]$	0.001^2	0.02^2
$\frac{\Delta p}{\Delta t}$ $[\mathrm{m}/\Delta t]$	0	0.05

Table 7.1. Varying covariance of process noise

The equation for calculating the process noise covariance is given below. If $\frac{\Delta p}{\Delta t}$ is larger than 0.05 the process noise covariance is set to 0.02^2.

$$Q = Q_{min} + \left(\frac{Q_{max} - Q_{min}}{0.05} \right) \frac{\Delta p}{\Delta t} \tag{7.3}$$

The event studied in the validation in chapter 5 is also studied in this chapter. The Kalman filter estimate of the velocity is transformed to a flow rate by multiplying with the flow area.

As initial conditions the analysis state variable (velocity) is set to zero, and the error on the analysis estimate is set to 0.001^2 corresponding to Q_{min}.

In figure 7.4 the Kalman filter estimate is plotted with the output from the flow sensor.

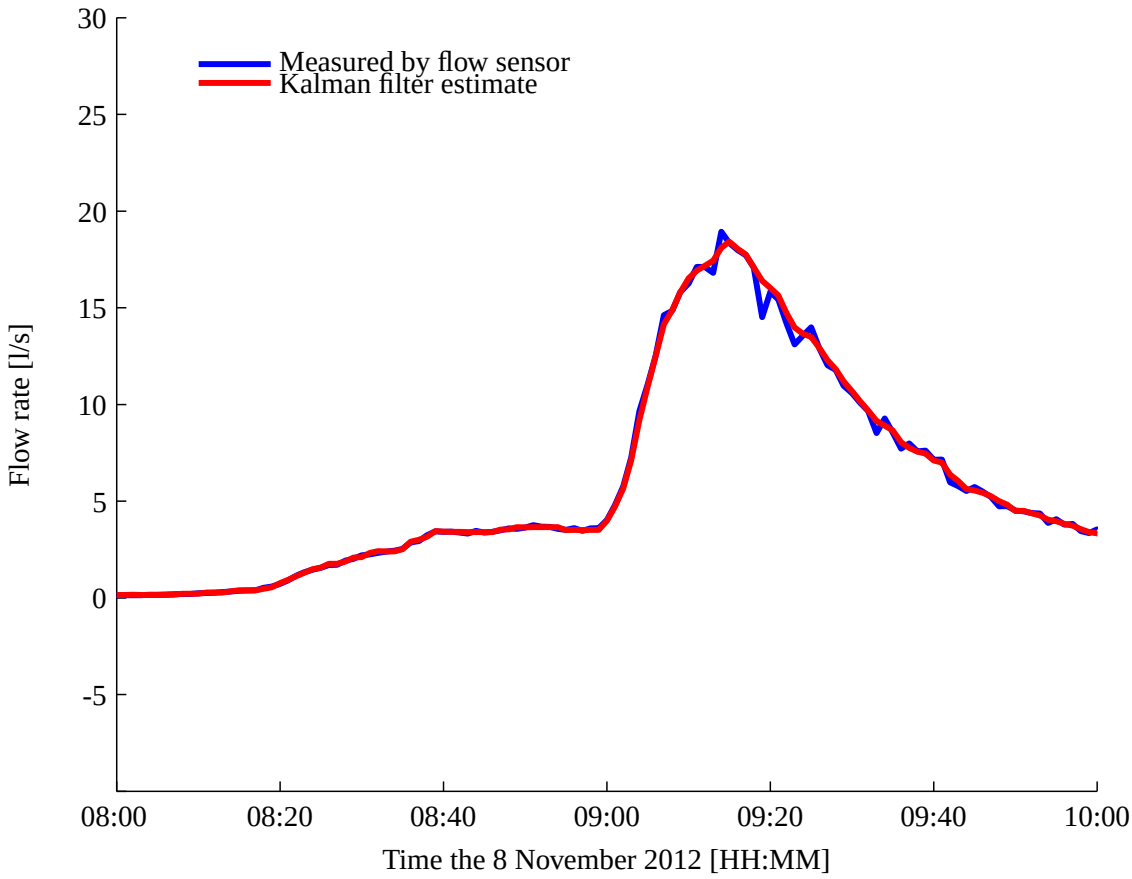


Figure 7.4. Measured and Kalman filter estimate

The Kalman filter estimate is close to the measured, although it seems as some noise have been removed. The effect of the noise removal is minimal in this case, however the effect will increase in a system with larger pipes, where a small correction of the velocity will have a larger effect due to the increased flow area.

If for instance the noise on the velocity estimate where minimised by a moving average filter, then peak values would be removed due to the smoothing effect. This is not necessary the case in the method presented here. Because if both the velocity measurement and the system model indicate the velocity is increased to a certain level, then this will be the estimate from the Kalman filter.

It is recommended to use the method in case an imprecise velocity sensor is used. In appendix D.3 the raw MATLAB code used in this chapter is presented.

7.2 Summary

In this chapter a method for assimilation of a velocity- and a pressure sensor have been presented. The method is based on the Kalman filter. The requirement for the defined system model is that a V-h-relation have been calibrated. The chapter also revealed that it is a good assumption, to assume errors on velocity measurements are normally distributed.

The simple definition of the system model make it easy two implement it directly in the software of a flow sensor.

The Kalman Filter for Assimilation of In-situ Sensors With MIKE URBAN 8

One of the advantages with the Kalman filter is that not all state variables have to be measured in order to be updated by measurements.

With a hydraulic model of a sewer system running in real-time, then the hydraulic model could describe the system in the Kalman filter, and in-situ sensors could be used to update the hydraulic model. Thereby the state of the entire sewer system would be estimated. This would be an advantages when RTC is conducted.

In Denmark MIKE URBAN is widely used to model how a sewer system behave under rainfall events, and therefore many sewer systems in Denmark are already digitalised. Therefore to develop an assimilation tool which is compatible with MIKE URBAN would be preferable.

With the MIKE URBAN pipe network model as system propagator in the Kalman filter, the overall assimilation procedure could be as given in figure 8.1.

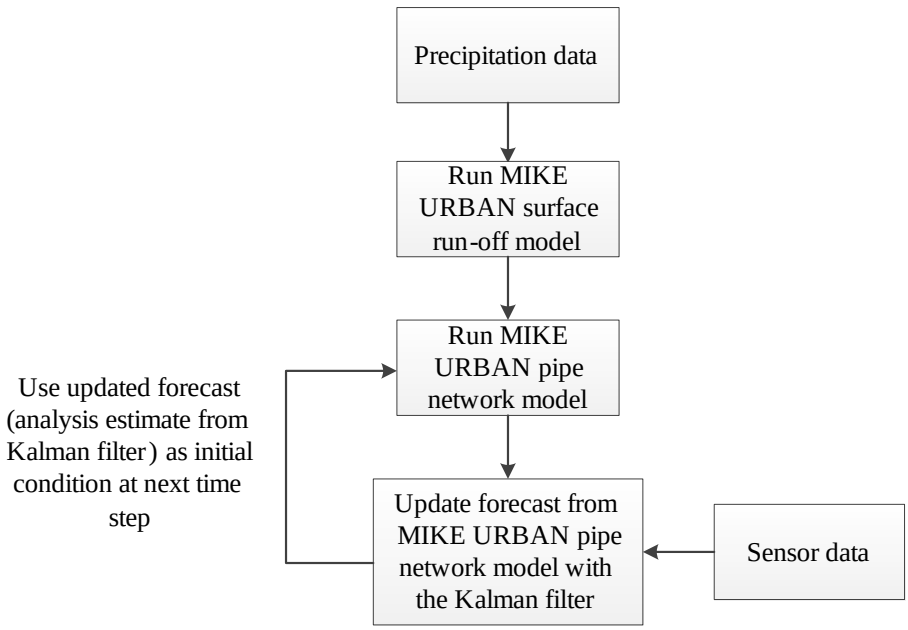


Figure 8.1. Assimilation procedure

Based on online precipitation data from for instance a rain gauge or a weather radar, the MIKE URBAN surface run-off model is used to estimate the boundary conditions for the MIKE URBAN pipe network model. The updated forecast from last time step is used as initial condition in the

pipe network model. The new forecast from the pipe network model is again updated by the Kalman filter with sensor measurements.

Since the Kalman filter is used to update the forecast, then the uncertainty on both the sensors and the model forecast is taken into account. However since the MIKE URBAN pipe network model is based on a solution of the Saint Vernaint equations the linear Kalman filter algorithm can not be used. When the system model is non linear then the error covariance cannot be propagated in time by the Kalman filter algorithm.

However since many problems cannot be described by a linear system model, different methods to overcome this problem have already been developed. This have led to the evolution of new editions of the Kalman filter. In this report an edition called the Ensemble Kalman filter (ENKF) will be used.

8.1 The Ensemble Kalman Filter

The ENKF was proposed in 1994 by the Norwegian Geir Evensen [Evensen, 1994].

As mentioned above the essential problem with the non linear system model is that the error covariance cannot be propagated analytically in time. The ENKF deals with this problem by introducing an ensemble of state variables.

An ensemble of state variables means that different estimates of the state variables are present. For instance if the ensemble contains 10 members, then 10 different estimates of the state variables will be present. The difference between the ensemble members are used to determine the error statistics.

8.1.1 The Theoretical Foundation

Referring to chapter 7 the error covariance was defined as given in equation 8.1, with respect to the forecast of the state variables and in equation 8.2 the error covariance is given with respect to the analysis estimate of the state variables.

$$P^f = E[(x^f - x)(x^f - x)^T] \tag{8.1}$$

$$P^a = E[(x^a - x)(x^a - x)^T] \tag{8.2}$$

[Evensen, 2003]

If an ensemble of state variables is present with N ensemble members, then the ensemble mean can be calculated according to equation 8.3.

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^n \tag{8.3}$$

N	Size of the ensemble	[-]
n	Index referring to the ensemble member	[-]

[Gillijns *et al.*, 2006]

The ensemble mean is interpreted as the best estimate of the state variables. The spread of the ensemble members, around the mean of the ensemble, is interpreted as the error between the true state and the estimate of the state variables. Consequently the error covariance matrix of the state variables, can be estimated according to equation 8.4 with respect to the forecast and according to equation 8.5 with respect to the analysis.

$$P^f \approx P_e^f = \overline{(x^{f,n} - \overline{x^f})(x^{f,n} - \overline{x^f})^T} \quad for\ n = 1,...,N \tag{8.4}$$

$$P^a \approx P_e^a = \overline{(x^{a,n} - \overline{x^a})(x^{a,n} - \overline{x^a})^T} \quad for\ n = 1,...,N \tag{8.5}$$

[Evensen, 2003]

The overline in equation 8.5 and 8.4 denotes the average.

The ensemble matrix is defined according to equation 8.6.

$$X = (x^{n=1},...,x^{n=N}) \tag{8.6}$$

Based on the mean of the ensemble and the ensemble matrix, the ensemble perturbation matrix $X^{'}$ is defined according to equation 8.7.

$$X^{'} = (x^{n=1} - \overline{x}\ ,..., x^N - \overline{x}) \tag{8.7}$$

Thereby the error covariance matrix can be calculated by equation 8.8.

$$P = \frac{X^{'}X^{'\ T}}{N-1} \tag{8.8}$$

The error covariance can be calculated both with respect to the forecast ensemble and the analysis ensemble by equation 8.8.

8.2 The ENKF With MIKE URBAN as System Model

The boundary condition for the pipe network model, would be the result from the surface run-off model. The surface run-off estimate will be corrupted by noise since uncertainty is associated with the measurement of the precipitation, the catchment description and the chosen surface run-off model. The distribution of the noise need to be known.

The noise from the boundary condition will be dispersed through the pipe network model, however since the system model is non linear, it cannot be analytically determined how the noise will be dispersed. Therefore the dispersion of the noise is modelled through Monte Carlo simulations, by adding random noise to the deterministic surface run-off estimate for each individual ensemble member. The random noise should be sampled from the distribution of the boundary noise. The noise on the forecasted state variables, can then be calculated based on the forecasted ensemble - see figure 8.2.

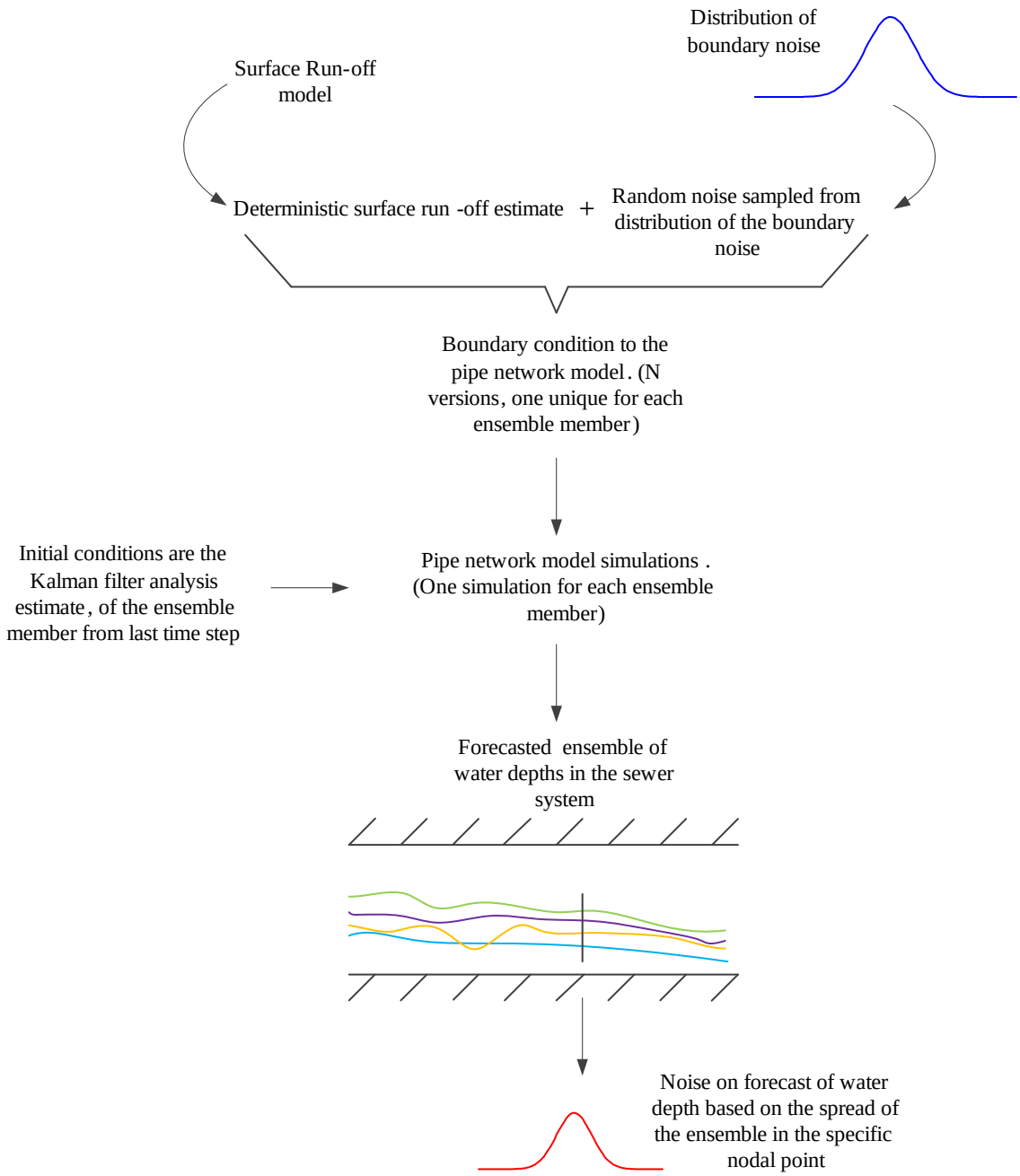


Figure 8.2. Flow chart of Monte Carlo simulations used to propagate error statistics of the forecast

Process noise due to numerical errors and an incomplete description of the sewer system would also be present in reality. Although these error sources are assumed to be neglectable, compared to the process noise from the surface run-off estimate.

To sum up the above, then in the forecast step each ensemble member are simply forecasted. The error statistics which can be derived from the forecasted ensemble is used in the next step to calculate the Kalman gain.

Recall the equation for the Kalman gain as given in chapter 6.

$$K_k = P_k^f H^T (H P_k^f H^T + R)^{-1} \tag{8.9}$$

H	Transition matrix which link state variables to measurements	[-]
R	Covariance of measurement noise	[-]
k	Time index	[-]
T	Matrix is transposed	[-]
f	Index referring to forecast	[-]

With the definitions given of the ensemble matrices the Kalman gain can be calculated according to equation 8.10.

$$K_k = P_k^f H^T (H P_k^f H^T + R)^{-1} = X_k^{'f} X_k^{'f\ T} H^T (H X_k^{'f} X_k^{'f\ T} H^T + Z_k^{'} Z_k^{'\ T})^{-1} \tag{8.10}$$

The variable Z represents the ensemble of measurements. The ensemble are generated according to equation 8.11.

$$z^n = z + v^n \quad for\ n = 1,...,N \tag{8.11}$$

Each member of the ensemble is a vector with size m, corresponding to the number of measurements. In equation 8.11, v is random measurement noise sampled from R as defined in chapter 7. As for the state variables, is an observation ensemble perturbation matrix $Z^{'}$ also defined, see equation 8.7.

Last step is to calculate the analysis estimate of the state variables. This is done for each ensemble member, because the analysis estimate of each ensemble member are used as initial conditions in next time step. The result in an updated ensemble matrix X_k^a.

$$X_k^a = X_k^f + K_k (Z - H X_k^f) \tag{8.12}$$

The updated ensemble are the initial conditions in the pipe network model at next time step.

8.2.1 Practical Implementation

The practical implementation require that initial conditions, corresponding to the ENKF analysis estimate from last time step, can be defined in the MIKE URBAN pipe network model. Furthermore is it a requirement that the MIKE URBAN surface run-off model can be launched where the simulation was ended at last time step.

How this can be done will not be investigated further in this master thesis. Instead the actual programming of the ENKF algorithm and the performance will be reviewed through a theoretical study. In order to conduct the theoretical study, a 1D diffusive wave model is programmed in MATLAB. Details about the diffusive wave model can be seen in appendix C. Thereby full access to manipulate the simulation at each time step is possible.

The programming necessary to configure the ENKF with MIKE URBAN would in general be the same. The 1D diffusive wave model should just be substituted with the MIKE URBAN pipe network model as system model.

8.3 The ENKF With a 1D Diffusive Wave Model as System Model

For simplicity the theoretical study is conducted on an one dimensional open channel flow, where the boundary conditions are completely determining the flow see figure 8.3.

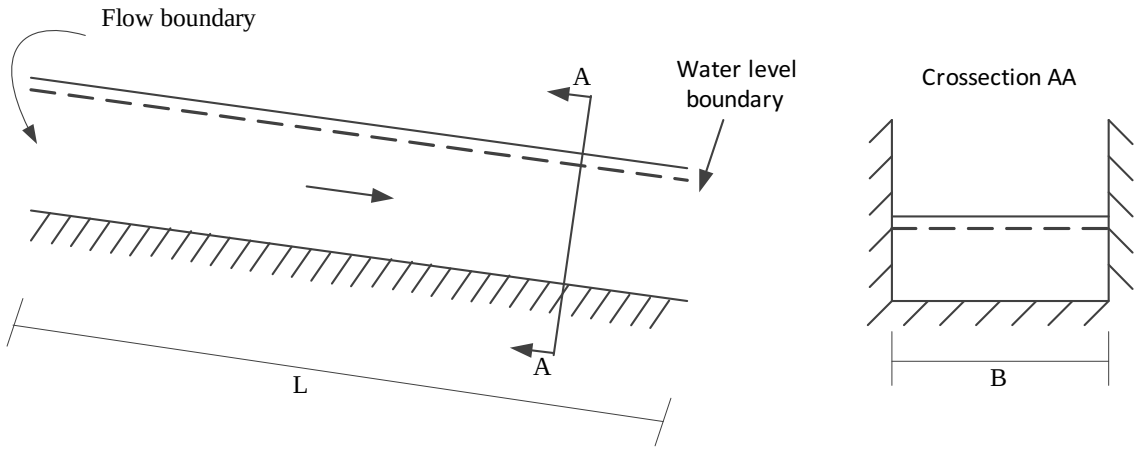


Figure 8.3. The flow scenario studied in the theoretical example

8.3.1 Assumptions for The Theoretical Study

It is assumed that the length of the channel L is 100 meter, and the width B is 0.5 meter. The bottom slope is set to 0.005 $^0/_{00}$, and the Mannings number is set to 70 $m^{1/3}/s$. The spatial discretisation is set to 1 meter resulting in 100 numerical nodal points, and the temporal discretisation is set to 0.01 seconds. It is decided that the water depth in all nodes are the state variables.

An ensemble with 50 ensemble members is used to propagate the error statistics. The system is modelled 150 seconds forward in time, and every 10th second is the ensemble updated by sensor measurements. It is assumed the water depth is measured in node 25 and 75. It is furthermore assumed that the noise on the sensors are Gaussian with zero mean and that the standard deviation is known.

As an initial state of the system, it is assumed that the water depth is 0.2 meter in all nodal points. Although each ensemble member will have their own initial state generated by adding random Gaussian noise with zero mean and a standard deviation corresponding to 0.01 meter.

If the same initial conditions were used for all ensemble members, then the ENKF would interpreted this as a perfect model forecast, and sensor measurements would not influence the model before the ensemble starts to diverge.

The lower boundary condition is a water level boundary. The boundary is set to be equal to the water level in the nodal point just upstream.

It is assumed that the upper flow boundary have been calculated based on a deterministic surface run-off model. The deterministic flow boundary condition for the 15 time steps are given in table 8.1

Time [s]	Flow [m^3/s]	Time [s]	Flow [m^3/s]
0 - 10	0.13	80 -90	0.2
10 - 20	0.2	90 - 100	0.13
20 - 30	0.25	100 - 110	0.13
30 - 40	0.3	110 - 120	0.13
40 - 50	0.3	120 - 130	0.13
50 - 60	0.3	130 - 140	0.13
60 - 70	0.25	140 -150	0.13
70 - 80	0.25	-	-

Table 8.1. Flow boundary used in theoretical example

Although we are aware that the deterministic estimates are not precise since there are uncertainties related to the surface run-off model. However knowledge about the statistical properties of the noise on the flow boundary is known.

The standard deviation of the noise on the flow boundary and the measurement noise is given in table 8.2.

	Noise on flow boundary [m^3/s]	Measurement noise [m]
Standard deviation	0.005	0.001

Table 8.2. Noise on flow boundary and measurement noise. Both sensors are subject to the same measurement noise

For each time step the ensemble members would have their own boundary condition due to the random noise.

The matrix H in the ENKF is formed based on the assumptions defined above. The matrix links the state variables x to the measurements z, and is defined according to the output equation 8.13.

$$z_k = Hx_k + v_k \tag{8.13}$$

x is a column vector with 100 entries (representing the state variables) and z is a column vector with 2 entries (representing the measurements). H is then a matrix with size $2x100$, where all entrances will be zero, except $H(1,25)$ and $H(2,75)$ which will be equal to one. These entrances will be one because it is the state variables in node 25 and 75 which are measured, and the measurements are the same physical property as the state variables. v is random measurement noise. The output equation is important because it describes how the model is linked to the measurements. The equation states that the sensors measure the true water depth plus some random noise.

Since it is a theoretical study no measurements of the state variables in node 25 and 75 are present in reality. Therefore for this theoretical study synthetic measurements in node 25 and 75 are generated.

In order to do this a model representing the true water depth is modelled. This model will in the following be named the truth model. The truth model is basically modelled as the other ensemble members, but the truth model is not subject to any assimilation.

This truth model is what is desired to estimate based on the 50 ensemble members and the measurements which will give an indication of what the true water depth is in node 25 and 75. With the truth model the measurements can be generated by equation 8.13. For the measurement generation x represents the state variables from the truth model. The process is described in figure 8.4.

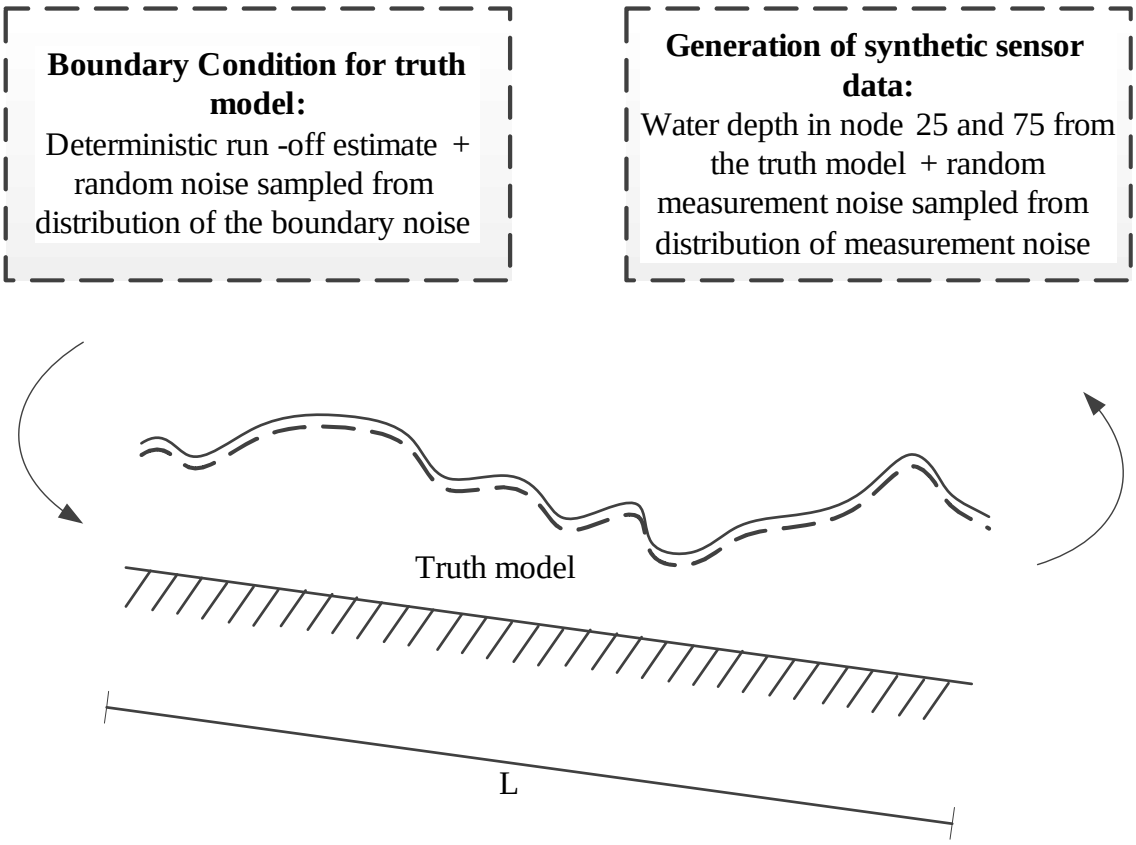


Figure 8.4. The truth model and generation of synthetic sensor data

With a truth model the performance of the ENKF can also be evaluated with respect to this truth model.

8.3.2 Results

At first the result from the truth model is studied. In figure 8.5 is the water depth plotted at 3 different time steps. At each of these time steps is an ensemble of the water depth also present. If the forecast ensemble is studied in the same time steps, it would look as given in figure 8.6

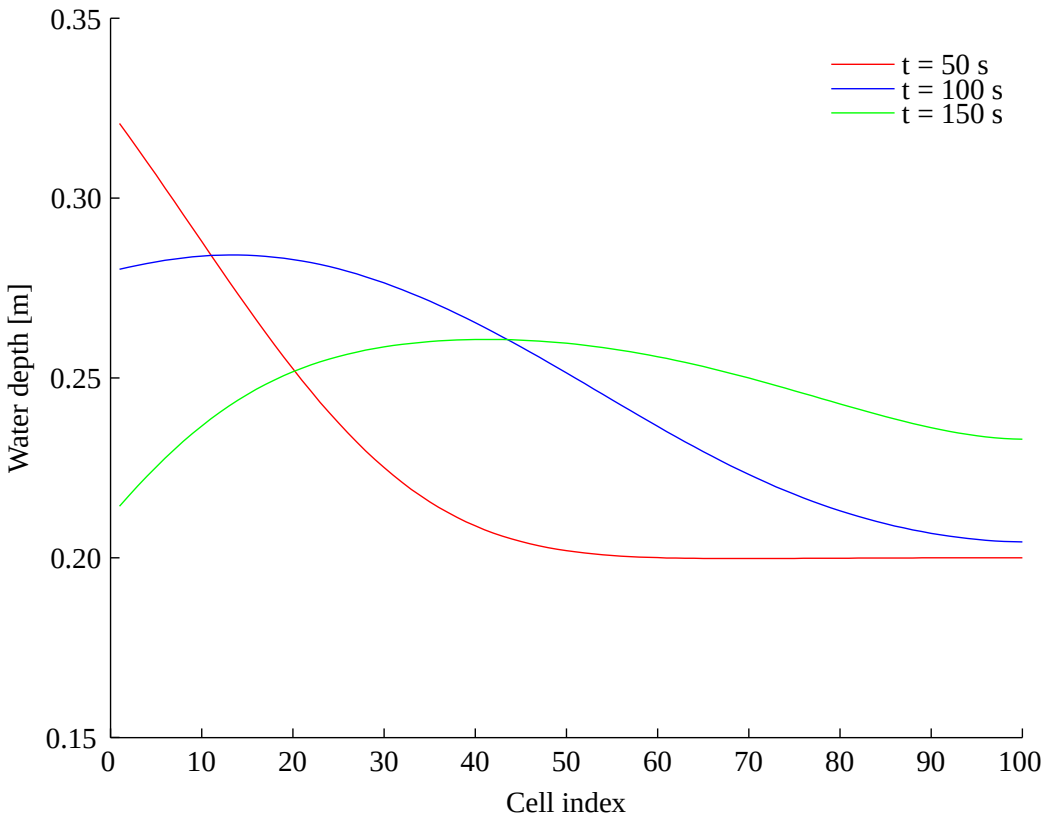


Figure 8.5. Water depth extracted from the truth model at three different time steps

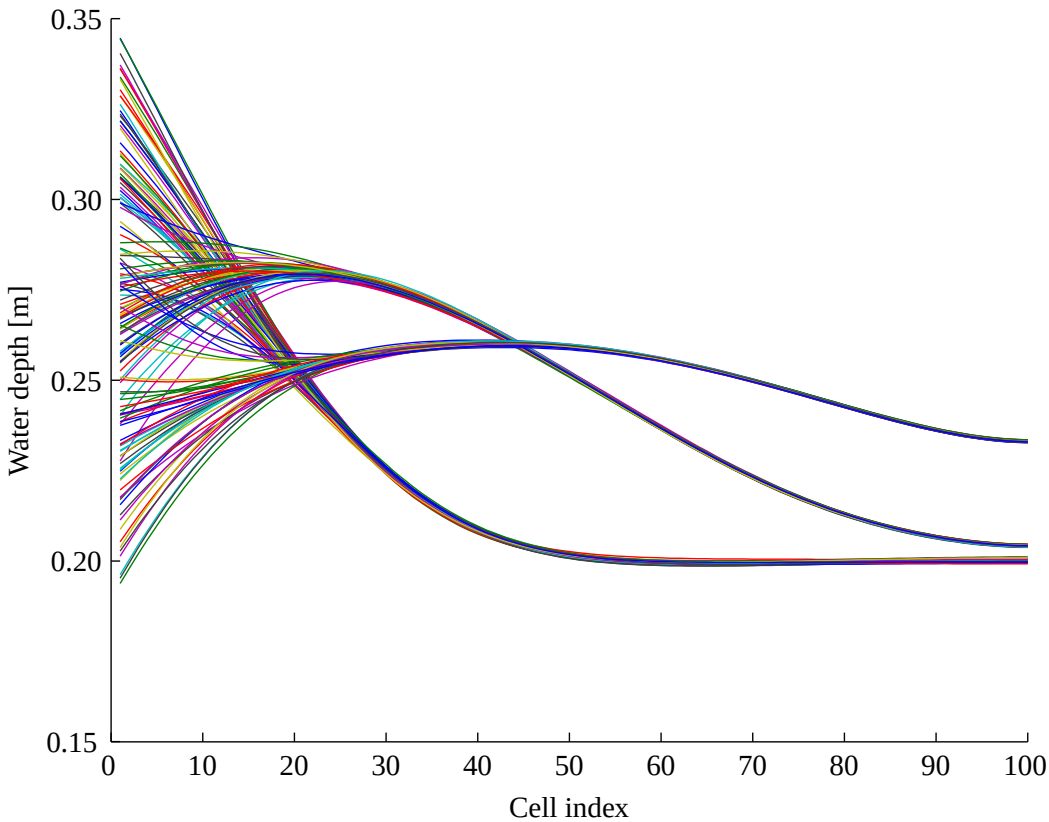


Figure 8.6. Ensemble of Water depth at time steps corresponding to time steps in figure 8.5

It is clear from the ensemble that the uncertainty is largest close to the flow boundary. Remember the uncertainty is interpreted as the spreading around the mean of the ensemble. This is as expected, because further downstream all ensemble members have been corrected through a number of data assimilation cycles. Furthermore will an error disperse to surrounding cells, as a consequence of wave diffusion.

In order to investigate if the assimilation of the measurements enhance the estimated state of the system, then a model where no random noise was added to the boundary condition where also launched. Then this deterministic approach was compared to the ENKF estimate with respect to the truth model. In figure 8.7, the results are compared after 150 s.

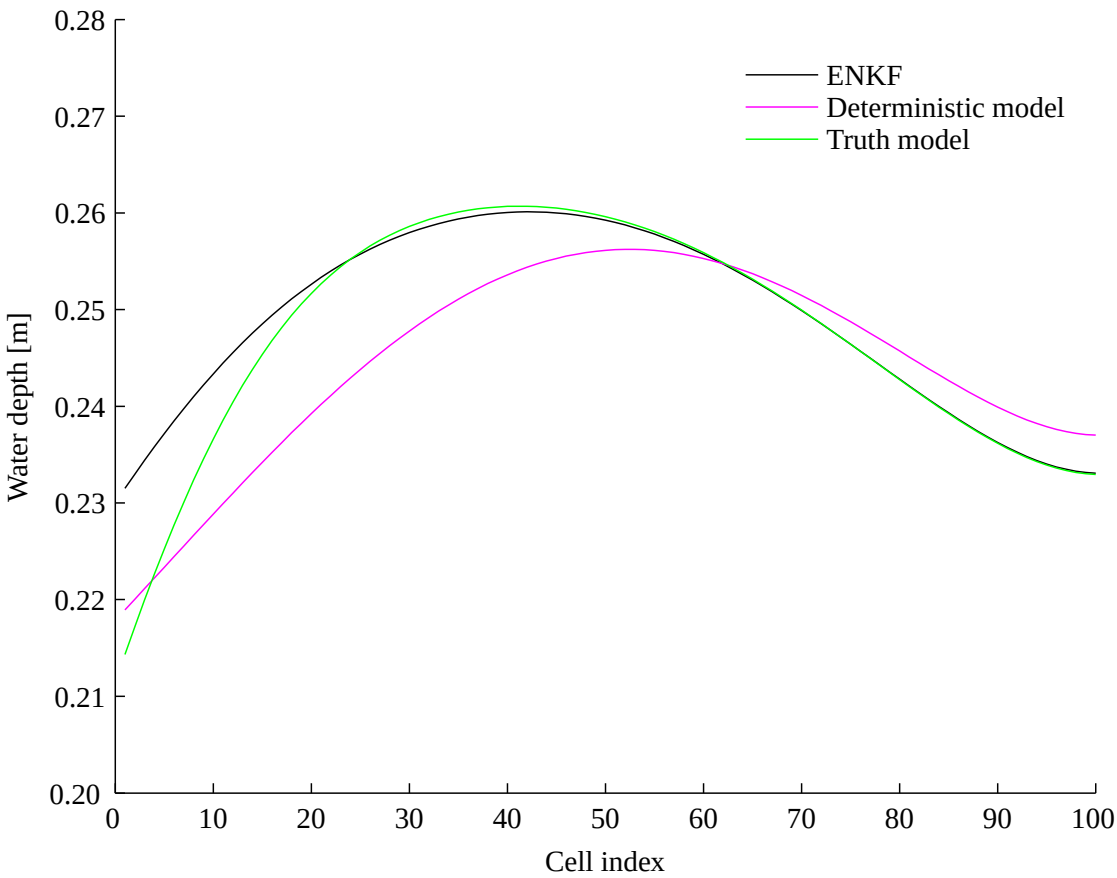


Figure 8.7. Comparison of ENKF and deterministic model with truth model after 150 s

From figure 8.7 it can be seen, that the ENKF estimate is close to the truth model compared to the result obtained from the deterministic model. In figure 8.8 the performance of the ENKF is evaluated by calculating the mean squared error (MSE) at each time step. The MSE is calculated according to equation 8.14.

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(\hat{x_n} - x_n)^2 \tag{8.14}$$

$\hat{x}$	Vector with estimates of the water depth	$[-]$
x	Vector with water depths from the truth model	$[-]$
N	Total number of state variables	$[-]$
n	Index referring to computational nodes	$[-]$

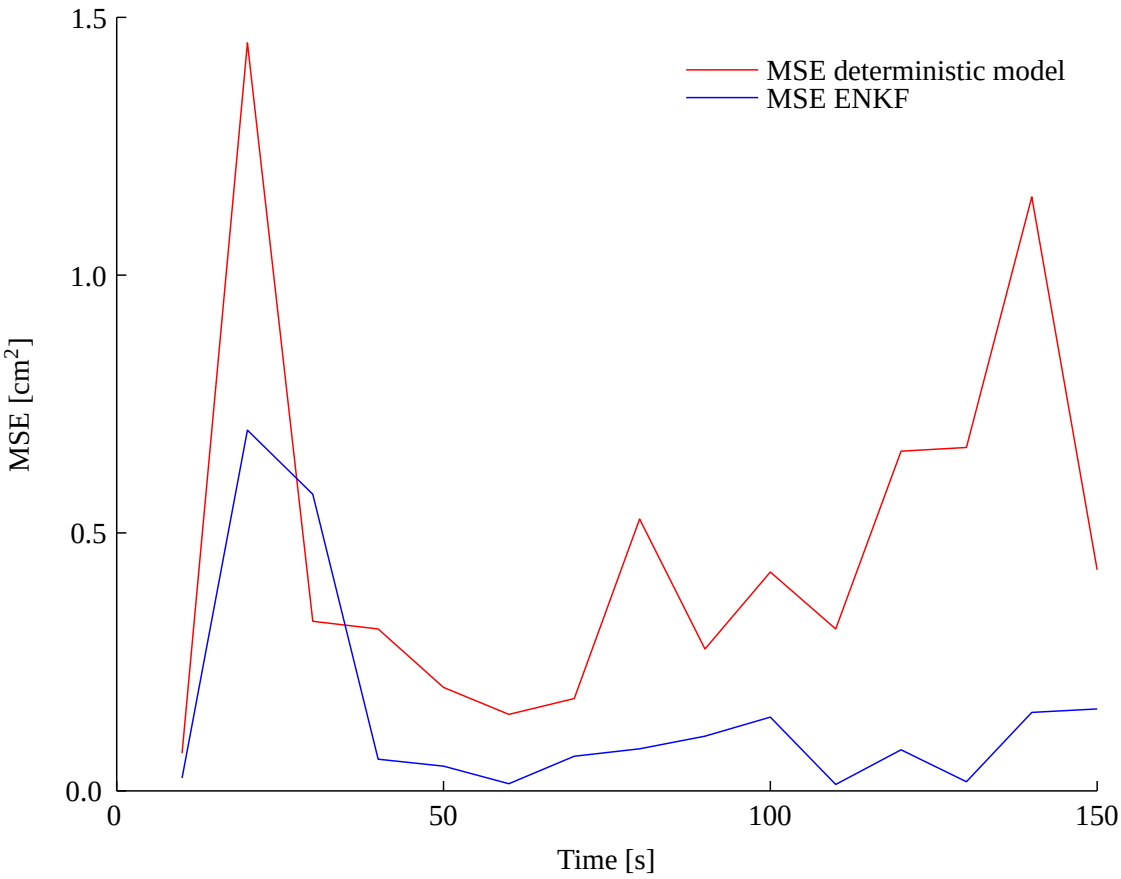


Figure 8.8. Performance of the ENKF

Based on the results it seems as the ENKF performance is satisfying compared to the deterministic approach. The MATLAB code for the ENKF and the diffusive wave model can be seen in appendix D.4.

8.4 Summary

In this chapter the possibility of assimilating in-situ sensors with the MIKE URBAN pipe network model was explored. The motivation was to enhance the knowledge about the true state of the sewer system. The potential of the ENKF for this purpose was revealed by conducting a theoretical study on simple 1D diffusive wave model. In practice the analysis part where the ENKF is used

to update the model would not be different if the MIKE URBAN pipe network model was used to model the system.

The potential of using the ENKF to assimilate in-situ sensors with the MIKE URBAN pipe network model is huge. Various type of sensors can be assimilated with the model for instance water level sensors, flow sensors, overflow registrations, pump registrations etc. This will improve the data foundation for RTC, since all sensors are taken into account in order to estimate the state of the sewer system.

What limits the practical implementation is the computational burden of modelling a large ensemble. At least it would require serious computational power to run the assimilation in real-time on a large MIKE URBAN model.

In some situations it might be possible to reduce the computational burden. The reason for modelling an ensemble are to propagate the error covariance which is used to calculate the Kalman gain. It is likely to believe that the Kalman gain will become constant in time in many practical situations. If for instance figure 8.6 is examined, it can be seen that the uncertainty on the forecast does not seem to change remarkable in time. If the Kalman gain does not change in time there is no need for modelling the ensemble. This topic will be discussed further in the discussion in chapter 10.

Conclusion 9

In this master thesis the aim was to improve the data for RTC of sewer systems. In order to achieve the objective it have been investigated how information about the true state of a sewer system can be improved. Information about the state of a sewer system is vital in order to conduct RTC.

In the first part of the master thesis a method for real-time validation of in-situ sensors was presented. In order to test the developed method in-situ sensors was installed in a sewer system in Aarhus. The acquired data from the in-situ sensors was subject to different kind of errors.

The developed validation method proved to be successful in identifying the errors on the acquired data. Furthermore when sensors in fail mode was detected, the developed method was capable of substituting the missing data by using redundant sensors.

The developed method is suitable for monitoring of in-situ sensors on a smaller catchment. However in case real-time control is conducted on large catchment where many sensors are installed it will require to much time to calibrate the developed method. This is due to the simplified flow model which have to be calibrated uniquely for the specific area and with respect to the specific sensors which are installed.

When real-time control is conducted on a large catchment it will be recommended to use the ENKF to assimilate in-situ sensors with a hydraulic model of the sewer system. Because then the state of the entire sewer system is estimated and the uncertainty on both the in-situ sensors and the model forecast are taken into account.

However it is still recommended to use simple control algorithms to detect if the in-situ sensor measurements for instance are out of physical range. Because if these non physical values are used as measurement input to the ENKF they will only make the estimated state more error-prone.

Through a theoretical study it was proven that the ENKF estimates the true water depth in an open channel, better then a deterministic model where no data assimilation is used. Although since the study was conducted on theoretical data the noise on both the boundary conditions and the fictive sensors was known.

If the ENKF is used with wrong assumptions about the noise distributions the theoretical fundamental is lost. It is therefore very important to estimate both the noise on the measurements and the noise on the boundary conditions accurate when the ENKF is applied.

Overall this master thesis have presented validation and assimilation techniques which can improve the data for RTC of urban sewer systems.

Discussion 10

Assimilation of in-situ sensors with a hydraulic model has a huge potential for improving data for RTC of urban sewer systems. When the ENKF is used for the assimilation the uncertainty is minimised and the state of the entire sewer system is estimated. Although there are one primary obstacle, which make it difficult to implement the ENKF with the MIKE URBAN pipe network model as system model, the computational burden. Unless the computational burden is minimised it will not be implemented in real life scenarios in the near future.

The computational burden is due to the ensemble which has to be modelled. The computational time is proportional to the size of the ensemble. If the ENKF is used it is therefore important to identify the ensemble size which is necessary to propagate the error statistics essentially used to calculate the Kalman gain.

The necessary size of the ensemble could be identified by conducting a sensitivity analysis. In the sensitivity analysis different rainfall events should be studied. Then for each studied rainfall event the Kalman gain as a function of the ensemble size should be evaluated. The objective would then be to identify the smallest possible ensemble. The necessary ensemble size might be different between rainfall events.

However if the ensemble size is reduced to for instance 10, the computational burden would still be to large for practical implementation when modelling large urban sewer systems.

As mentioned in chapter 8 it might be possible to use a constant Kalman gain in some applications. This will correspond to a steady state Kalman filter. If this is the case there is no reason for modelling the ensemble and the only calculation necessary from the Kalman filter is the equation for calculating the analysis estimate of the state variables.

It could be revealed, if it is a good assumption to assume a constant Kalman gain, by modelling a proper ensemble size for different rainfall events, and then evaluating the Kalman gain as a function of time. If the study reveal that a constant Kalman gain cannot be used for all rainfall events, then it might be an idea to generate a Kalman gain archive. In this archive different Kalman gain corresponding to different rainfall events could be stored, and then when the analysis estimate is calculated a Kalman gain is loaded from the archive.

Another perspective is to base the MIKE URBAN simulations on precipitation forecasts. The precipitation forecast could for instance be generated by weather radar observations. For instance could a forecast with a lead time on 10 minutes be used. Then the ensemble could be modelled 10 minutes forward in time, and when the measurements from the in-situ sensors are acquired 10 minutes later the forecast is updated. This save time since less assimilation cycles have to be conducted.

There is also another possibility of using simplified models to forecast the hydraulics of the sewer system and thereby calculation time can be reduced and the ensemble can be modelled.

As presented above there is different possibilities which could be explored in order to minimise the computational time. Although if wrong assumptions about the noise on the input to the MIKE URBAN pipe network model is used, then modelling the ensemble get meaningless. Because in this case a lot of computational effort is spend on propagating error statistics which are wrong in the first place.

If the ensemble is modelled it is therefore important to estimate the uncertainty related to the precipitation input to the model. The uncertainty on the precipitation input could be described by an ensemble. Then this an ensemble could be used directly as precipitation input to the surface run-off model.

Finally it can be discussed how dependent the estimate of the state variables are on minor errors in the Kalman gain. For instance assume that a constant Kalman gain is used. Furthermore assume that the Kalman gain in general is to low compared to the true Kalman gain. This would result in a higher weighting of the model forecast, which means that the system will response at a slower rate to the measurements. As a consequence the innovation (ie. the difference between the model forecast and measurements) would increase. However since the update of the model forecast is equal to the innovation multiplied with the Kalman gain, then the error will not be self-reinforcing.

Bibliography

Kloak A/S Aalborg Forsyning. Vision 2100 - samt udviklingsplan frem til år 2025, November 2009.

Ulrik Andersen, **December 2012**. Kampen om kloakkerne stopper aldrig. URL http://ing.dk/artikel/135062-kampen-om-kloakkerne-stopper-aldrig. Last downloaded 08-01-2013.

Bilal M. Ayyub and Richard H. McCuen, **2003**. Probability, Statistics, and Reliability for Engineers and Scientists. CHAPMANN & HALL/CRC.

Corp. Banner Engineering, **2012**. Banner-U-Gage-QT5OU-Series-Sensors. Banner Engineering, Corp.

J. L. Bertrand-Krajewski, J. P. Bardin, M. Mourad, and Y. Béranger, **2003**. Accounting for sensor calibration, data validation, measurement and sampling uncertainties in monitoring urban drainage systems. Water Science and Technology, 47, 95–102.

M.V. Bijnen and H. Korving, **2008**. Application and results of automatic validation of sewer monitoring data.

N. Branisavljevic, D. Prodanovic, and D. Pavlovic, **2010**. Automatic, semi-automatic and manual validation of urban drainage data. Water Science and Technology, page 9.

Michael Brorsen and Torben Larsen, **2009**. Lærebog i HYDRAULIK. Aalborg Universitetsforlag.

DHI, **2012**. DHI MATLAB toolbox. URL http://www.dhisoftware.com/Download/DocumentsAndTools/Tools/DHIMatLabToolbox.aspx.

DHI, **2011**. MIKE by DHI.

DMI, **2012a**. URL http://www.dmi.dk. Last Downloaded 14-12-2012.

DMI, **2012b**. URL http://www.dmi.dk/dmi/index/erhverv/spildevandskomiteens_regnmaalersystem.htm. Last downloaded 07-11-2012.

C. Henry Edwards and David E. Penney, **2008**. CALCULUS EARLY TRANCENDENTALS. Pearson, 7 edition.

Environmental Protection Agency, **2012**. Vandmiljøplanerne - et historisk overblik. URL http://www.mst.dk/Borger/Landbrug_og_miljo/vandmiljoplaner/vandmiljoplaner_overblik.htm. Last downloaded: 29-04-2013.

Environmental Protection Agency, **2011**. EU's vandrammedirektiv. URL http://www.naturstyrelsen.dk/Vandet/Vandplaner/EU-vandrammedirektiv/. Last downloaded: 29-04-2013.

Geir Evensen, **1994**. Sequential data assimilation with a nonlinear quasi-geostrophis model using Monte Carlo methods to forecast error statistics. JOURNAL OF GEOPHYSICAL RESEARCH, 99, 10,143–10,162.

Geir Evensen, **2003**. The ensemble Kalman Filter: theoretical formulation and practical implementation. Ocean Dynamics, 53, 343–367.

L. Fuchs and T. Beeneken, **2005**. Development and implementation of a real-time control strategy for the sewer system of the city of Vienna. Water Science & Technology, page 8.

S. Gillijns, O. Barrero Mendoza, J. Chandrasekar, B. L. R. De Moor, D. S. Bernstein, and Ridley, **2006**. What is the Ensemble Kalman Filter and How Well Does it Work?

H.J.Liefting and J.G. Langeveld, **2008**. Sewer Monitoring Projects: Data Collection, Data Handling and Data Quality in Arnhem.

Phil Kim, **2010**. Kalman Filter for Beginners with MATLAB Examples. A-JIN Publishing Company.

Rudolf Emil Kálmán, **1960**. A New Approach to Linear Filtering and Prediction Problems. ASME "Journal of Basic Engineering".

Emmanuel Kopecny, Stéphane Entem, Antoine Lahoud, Arne Moeller, Lars Yde, and Marc Soulier, **2001**. REAL TIME CONTROL OF THE SEWER SYSTEM OF BOULOGNE BILLANCOURT A CONTRIBUTION TO IMPROVING THE WATER QUALITY OF THE SEINE, DHI.

U.K. Maheepala, A.K. Takyi, and B.J.C Perera, **2001**. Hydrological data monitoring for urban stormwater drainage systems. Journal of Hydrology, page 16.

Mathworks, **2012**. URL `http://www.mathworks.se/products/matlab/`.

M. Mourad and J.L. Bertrand-Krajewski, **2002**. A method for automatic validation of long time series of data in urban hydrology. Water Science and Technology, page 8.

Maria Isabel Ribeiro, **2004**. Kalman and Extended Kalman Filters: Concept, Derivation and Properties, Institute for Systems and Robotics Lisboa PORTUGAL.

Raul Rojas, **2003**. THE KALMAN FILTER. URL `http://robocup.mi.fu-berlin.de/buch/kalman.pdf`. FREIE UNIVERSITÄT BERLIN, INSTITUT FÜR MATEMATIK.

Dan Simon, **2001**. Kalman Filtering. Embedded Systems Programming, 14, 72–79.

M.K. Stinson, **2005**. BENEFITS OF SEWERAGE SYSTEM REAL-TIME CONTROL. ASCE, page 11.

Siao Sun, Jean luc Bertrand-krajewski, Anders Lynggard-Jensen, Joep van den Broeke, Florian Edthofer, Maria do Céu Almeida, Álvaro Silva Ribeiro, and José Menaia, **2011**. Literature Review of Data Validation Methods, SEVENTH FRAMEWORK PROGRAMME EU.

Inc. Teledyne Isco, **2012**. Isco 2150 Area Velocity Flow Module. Teledyne Isco, Inc.

Greg Welch and Gary Bishop, **2006**. An Introduction to the Kalman Filter, Department of Computer Science, University of North Carolina.

Leif Winther, Jens Jørgen Linde, H. Thorkild Jensen, and Leo Lund Mathiasen, **2006**. Afløbsteknik. Polyteknisk forlag.

Flow Routing Method A

In order to investigate whether a constant delay is a good assumption, a theoretical study was conducted. For the theoretical study a simple model was created in MIKE URBAN. The model correspond to the set up in figure A.1.

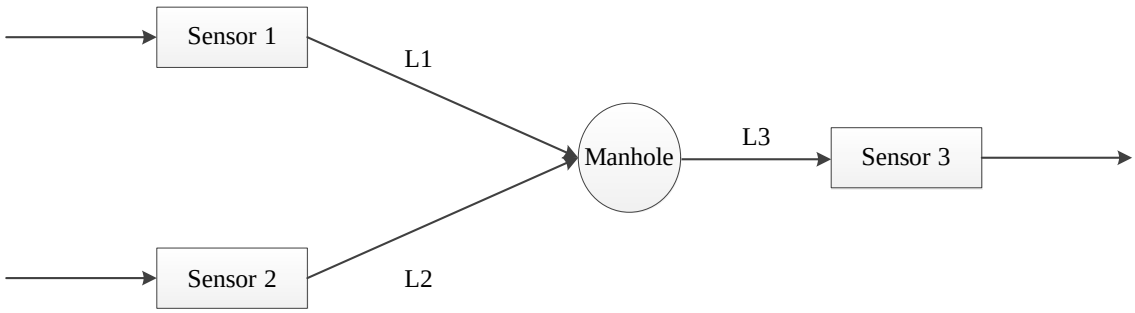


Figure A.1. Simple flow scenario

The Manning number was set to 75 $m^{\frac{1}{3}}/s$ for all links, and the resolution of the numerical grid was set to one meter. The model only covers the pipes between the sensors, and the manholes in relation to these pipes. Informations regarding the pipes are given in the table below.

Link	Slope [$^0/_{00}$]	Diameter [mm]	Length [m]
L1	4	350	43
L2	2	600	57
L3	1	600	114

Table A.1. Information about pipes used in the theoretical study

Imagine that no water is running in the pipe between sensor two and the manhole, and all water which reach sensor 3 originates from the manhole where sensor one is installed. In this case the delay between a measurement at sensor one, and sensor three can be calculated.

As an example a constant flow rate is applied at the manhole with sensor one for three minutes as boundary condition. The simulation is then launched and the flow rate is extracted at sensor 3. The delay is then calculated, as the time it takes 50 percent of the accumulated flow to reach sensor three. An example is given in figure A.2.

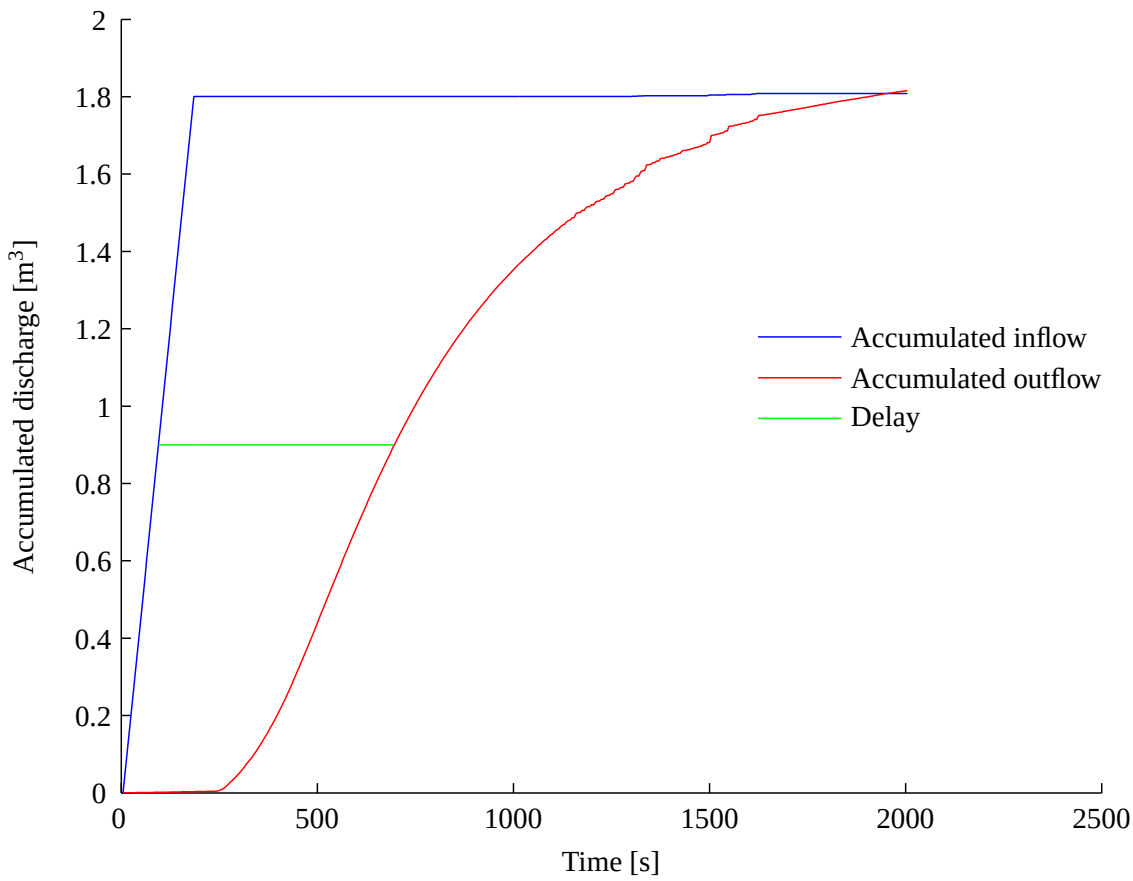


Figure A.2. Accumulated discharge at sensor one

In figure A.2 the boundary condition is set to 10 l/s - which gives a accumulated flow of 1.8 m^3 over the three minutes. The delay can then be calculated to 610 s with the described method. From figure A.2 it can also be seen that the flow rate change through the system, since the steepness of the two curves is not the same. However some of the this might be related to numerical errors when MIKE URBAN is solving the Saint Vernaint equations, although in theory a wave will flatten out through the system, due to the diffusive term in the Saint Vernaint equations.

In order to establish a relationship between the applied boundary condition and the delay, the model is launched with different boundary conditions. To save time a program is set up in MATLAB [Mathworks, 2012]. Thereby the MIKE URBAN engine (MOUSE) is controlled directly through MATLAB and the model do not have to be launched manually for each simulation. The raw MATLAB code can be seen in appendix D.1. The flow chart for the MATLAB program is given in figure A.3.

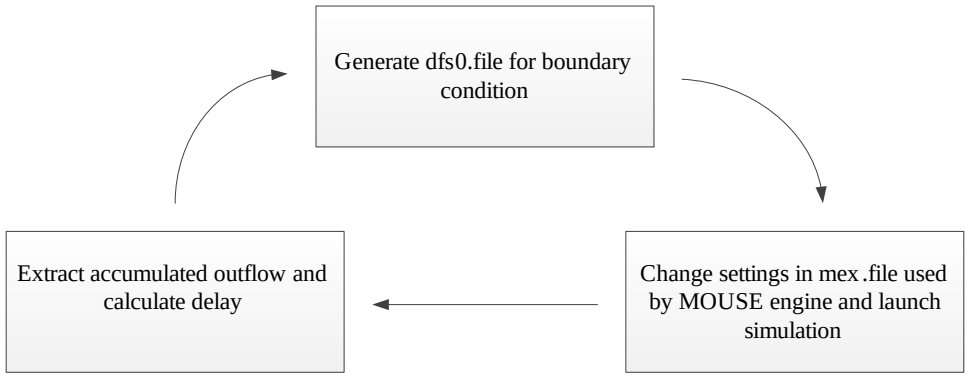


Figure A.3. Flowchart for MATLAB program

As boundary conditon the discharge was applied in an interval from 1-80 l/s, with a resolution of 1 l/s, ie. a total of 80 simulations were conducted, and at each simulation the boundary condition was applied for three minutes. Thereby the relationship in figure A.4 can be established.

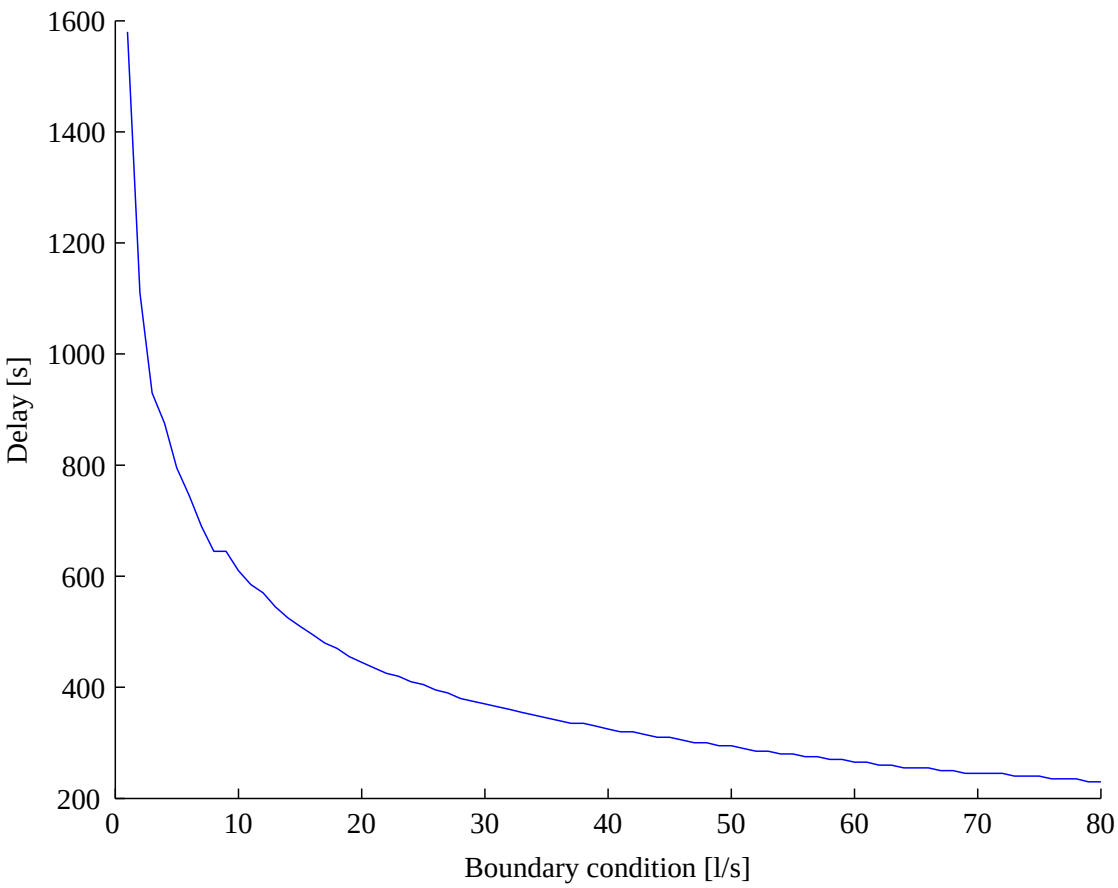


Figure A.4. Delay as a function of discharge at the boundary condition

Figure A.4 clearly indicate that a constant delay cannot be used, when data from the flow sensor and water level sensors should be compared.

Case Study B

B.1 System Configuration for Case Study

In table B.1 the upper threshold used in the initial control are given.

Sensor	Upper threshold	Unit
Flow sensor (Q)	0.25	$[m^3/s]$
Flow sensor (V)	1	$[m/s]$
Flow sensor (h)	1	$[m]$
Water level sensor 1	0.55	$[m]$
Water level sensor 2	0.99	$[m]$
Rain gauge	33.33	$[my - m/s]$

Table B.1. Upper threshold used in the initial control

B.2 Validation of Developed Method

In the figures below are the results form the second event studied in the validation of the developed method.

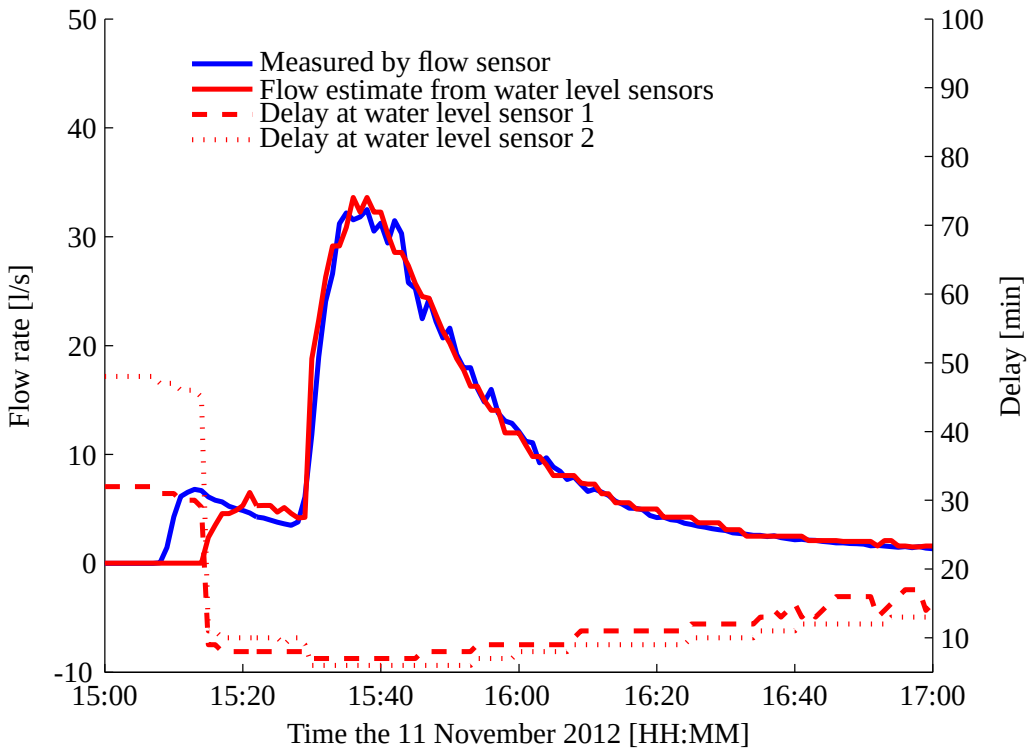


Figure B.1. Measured and estimated flow rate

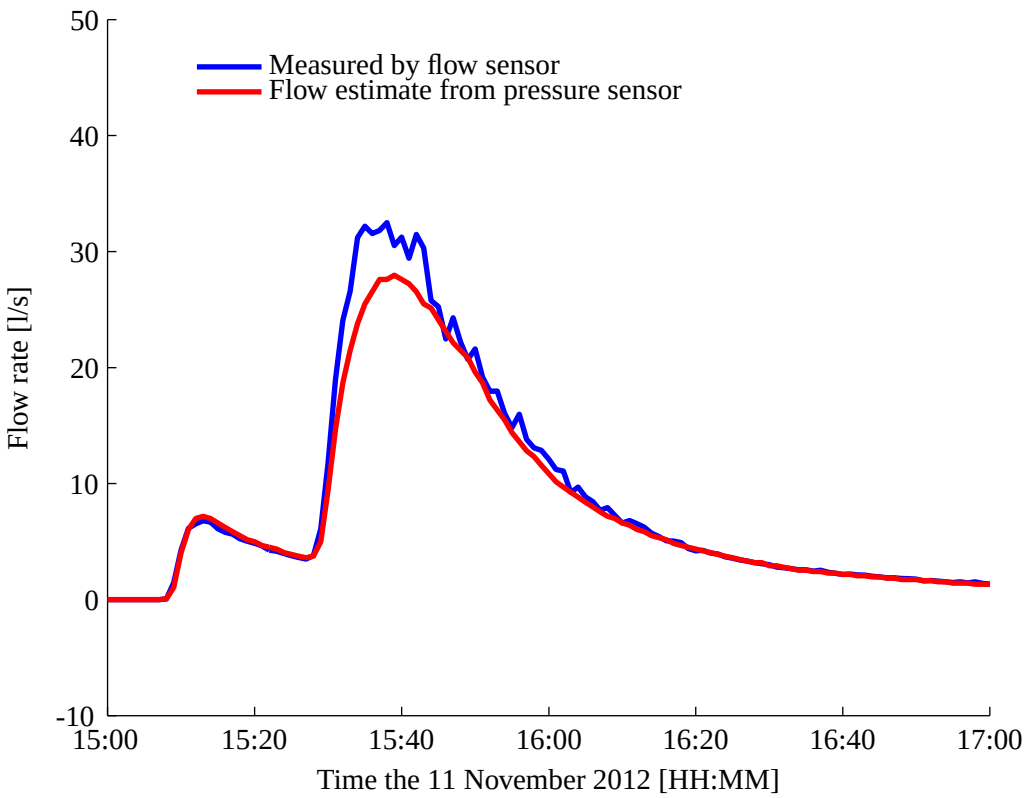


Figure B.2. Measured and estimated flow rate

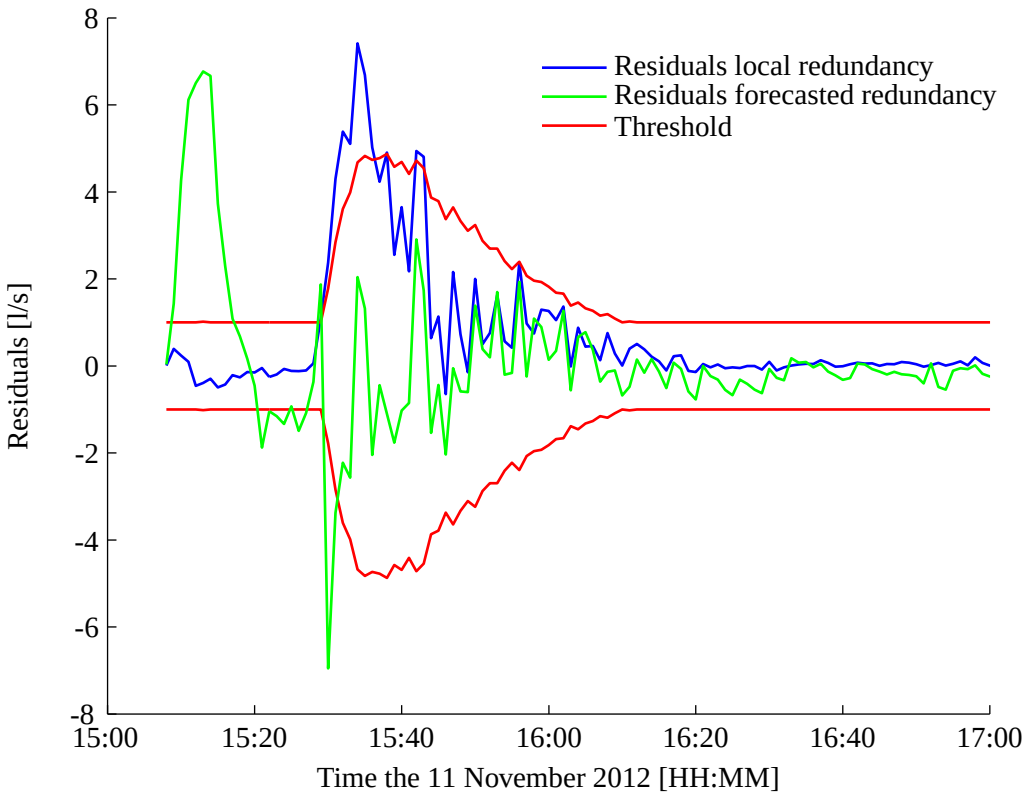


Figure B.3. Calculated residuals over the event

The Kalman Filter for Assimilation of In-situ Sensors With MIKE URBAN C

C.1 ENKF On a 1D Diffusive Wave Model

The diffusive wave model used in the theoretical example in chapter 8 is based on the continuity equation and Mannigs equation. Consider figure C.1.

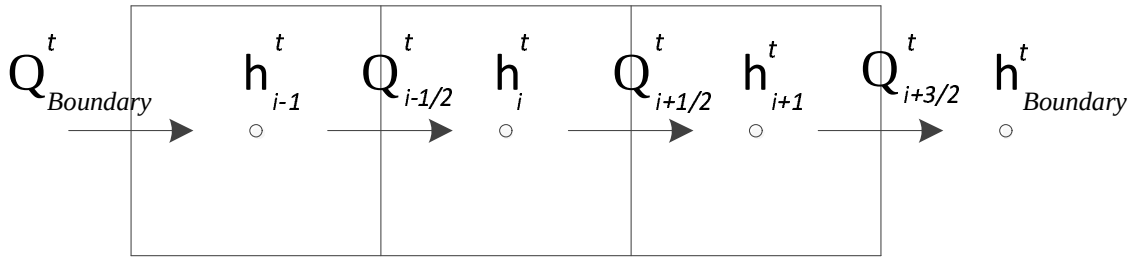


Figure C.1. Conceptual sketch of diffusive wave model. i refer to nodal index and t to time index

The figure illustrate a numerical model with three nodal points. In the nodal points the water depth is estimated, and in between the nodal points is the flow rate estimated. An explicit solution scheme is used to solve the system. By rewriting the continuity equation then the water depth in all nodal points can be calculated. In equation C.1 the update equation for the water depth in cell i at time t is given.

$$h_i^t = \frac{Q_{i-1/2}^{t-1} - Q_{i+1/2}^{t-1}}{dx\, b} dt + h_i^{t-1} \tag{C.1}$$

The equation is valid due to the rectangular cross section. The flow rate in between the nodal points are updated by Mannings equation, see C.2.

$$Q_{i+1/2}^t = m\, R(h_{i+1/2}^t, b)^{2/3}\, A(h_{i+1/2}^t, b)\, \sqrt{\frac{h_i^t + z_i - h_{i+1}^t + z_{i+1}}{dx}} \tag{C.2}$$

The variable z is the elevation of the bottom from a fixed reference.

MATLAB Programmes D

This appendix contains the raw program codes for the MATLAB programmes created during this master thesis.

D.1 Program for External Control of MIKE URBAN

In this section the program used to run MIKE URBAN through MATLAB is presented. In order to use the MATLAB code the DHI MATLAB toolbox have to be downloaded [DHI, 2012]. Furthermore is the code set up for a specific mex.file generated for the specific system which is analysed.

```
% Program MIKEUrbanMATLABControl
% Programmer: Daniel Brødbæk
% Master student, Aalborg University
% Project: Master thesis
%
% The Program creates a new dfs0 file to use as
% boundary condition. Afterward the program launch
% the MOUSE engine and finally extract results from
% the output file generated by the MOUSE engine.
% The program is setup for a specific mex.file generated
% by a MIKE URBAN model of the project case area.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Set parameters

% Clean up
close all
clear all
clc

% Add path to DHI MATLAB toolbox
addpath('mbin')

% Add some DHI NET Assemblies
NET.addAssembly('DHI.Generic.MikeZero.DFS');
NET.addAssembly('DHI.Generic.MikeZero.EUM');
import DHI.Generic.MikeZero.DFS.*;
import DHI.Generic.MikeZero.DFS.dfs123.*;
import DHI.Generic.MikeZero.*
```

```
30
31 % Define filename for new dfs0 file
32 FilenameDFS0 = 'DischargeFile.dfs0';
33 FilenameDFS0_2 = 'DischargeFile2.dfs0';
34
35 % Define startdata and timestep interval
36 StartTime=datenum(2011,01,01,00,00,00);
37 TimeStep=datenum(0000,00,00,00,00,01);
38
39 % Define filename for basis *.mex file
40 FilenameMEX = 'Simulation_HD.mex';
41
42 % Define basis simulation start and end strings in
43 % *.mex file:
44 % Look in *.mex file for the correct strings
45 SimulationStartStr_basis = 'Simulation_start';
46 SimulationEndStr_basis = 'Simulation_end ';
47
48 % Define basis dfs0 strings in *.mex file:
49 % Look in *.mex file for the correct strings
50 DFS0TSConnectionStr_basis = 'discharge.dfs0';
51 DFS0DataTypeName = 'Discharge';
52 DFS0TimeseriesName = 'Undefined';
53
54 DFS0StartStr_basis = '1900-01-01 00:00:00';
55 DFS0EndStr_basis = '2100-01-01 00:00:00';
56
57 % Paths to MOUSE engine
58 MOUSE_path =...
59 'C:\Program Files (x86)\DHI\2011\bin\MOUSESimLaunch.exe';
60
61 % Define data to search for in final result
62 % file M11.out.
63 % Look in M11.out file for the correct strings
64 Qaccin='Volume:   <Accumulated Inflow>';
65 Qaccout='Volume:   <Accumulated Outflow>';
66 % Display status
67 fprintf('\n Parameters set!\n');
68
69 %%
70 % Data for dfs0 file
71
72 % Allocation
73 Data=zeros(2000,1);
74 for i=1:80
75 DataTitle='Discharge time series';
```

```
% Define data properties
EumType='Discharge';     % Data type
EumUnit='m^3/s';         % Data unit

% Add constant discharge to first 3 minutes
Data(1:180,1)=i*0.001;

% Calculate end time on the basis for data length
% and timesteps
EndTime=StartTime+TimeStep*(size(Data,1)-1);

% Display status
fprintf('\n Data for dfs0 file generated!\n');

% New line
fprintf('\n');

%%
% Program code for generating dfs0 file

% Delete old DFS0 file
delete(FilenameDFS0)

% Create the file (Set to 1 if creating new file)
dfs0 = dfsTSO(FilenameDFS0,1);

% Set a file title
set(dfs0,'filetitle',DataTitle);

% Set startdate and timestep interval
set(dfs0,'startdate',datevec(StartTime));
set(dfs0,'timestep',datevec(TimeStep));

% Add number of timesteps
addTimesteps(dfs0,size(Data,1));

% Add Item to dfs0 file and assign EUM type and EUM units
% To list EUM types and units use the following codes:
% help listEumTypes
% help listEumUnits
% listEumTypes(dfsTSO())
addItem(dfs0,EumType,EumType,EumUnit);

% Add data to item 1 of the dfs0 file in one step
% Data is convert to single (floats) to avoid warnings.
```

```
% MIKE Zero assumes floats, MIKE URBAN handles both.
% Floats contains 7 degits and double contains
% 15 to 16 digits.
dfs0(1) = single(Data);

% Save and close file
save(dfs0);
close(dfs0);

% Display status
fprintf('\n DFS0 file created: ''%s''\n',FilenameDFS0);

%%
% Program code for reconfiguration of *.mex file
% for MIKE Urban batch run via MATLAB

% Read standard *.mex file
mex_basis = fileread(FilenameMEX);

% Copy mex basis to mex new
mex_new = mex_basis;

% Find and replace simulation start time so it fit
% the dfs0 file
% Start time is loacted in spaces from + 20 to + 38
SimulationStart_Index = regexp(mex_new,...
SimulationStartStr_basis);
for n=1:size(SimulationStart_Index,2)
    mex_new(SimulationStart_Index(n)+20:...
    SimulationStart_Index(n)+38)= datestr...
    (StartTime,'yyyy-mm-dd HH:MM:SS');
end

% Find and replace simulation end time so it fit
% the dfs0 file
% End time is loacted in spaces from + 20 to + 38
SimulationEnd_Index = regexp(mex_new,...
SimulationEndStr_basis);
for n=1:size(SimulationEnd_Index,2)
    mex_new(SimulationEnd_Index(n)+18:...
    SimulationEnd_Index(n)+36)...
    = datestr(EndTime,'yyyy-mm-dd HH:MM:SS');
end

% Find and replace DFS0 file parameters
mex_new = regexprep(mex_new,...
```

```
DFS0TSConnectionStr_basis, FilenameDFS0);
mex_new = regexprep(mex_new,...
DFS0DataTypeName, EumType);
mex_new = regexprep(mex_new,...
DFS0TimeseriesName, EumType);

mex_new = regexprep(mex_new, DFS0StartStr_basis,...
datestr(StartTime,'yyyy-mm-dd HH:MM:SS'));
mex_new = regexprep(mex_new, DFS0EndStr_basis,...
datestr(EndTime,'yyyy-mm-dd HH:MM:SS'));

% Write new *.mex content to file
% Filename for new file
FilenameMATLABMEX = 'MATLAB_run.mex';
% Open file
fid=fopen(FilenameMATLABMEX,'w+');
% Print content to file
fprintf(fid,'%s',mex_new);
% Close file
fclose(fid);

% Display status
fprintf('\n *.mex file generated!\n');

%%
% Program code for MIKE Urban batch run via MATLAB

% Display status
fprintf('\n MIKE Urban simulations started!\n');

% Start MOUSE engine start
dos(['"' MOUSE_path '" "' pwd '\' FilenameMATLABMEX...
    '" "HD" "Run" "Close" "NoPrompt" "-wait"']);

% Display status
fprintf('\n MIKE Urban simulations end!\n');

%%
% Extract data from MOUSE output file

% Use DHI program m11extra to read mouse output
% file (prf file)
!start /wait m11extra MATLAB_run.PRF

% The generated M11.out file is loaded
M11_new=fileread('M11.out');
```

```
% Define data which should be loaded by replacing
% zeros with ones
index=regexp(M11_new, Qaccin);
M11_new(index-8)='1';
index=regexp(M11_new, Qaccout);
M11_new(index-8)='1';

% Write M11_new to file
% Filename for new file
Filename = 'M11.in';
% Open file
fid=fopen(Filename,'w+');
% Print content to file
fprintf(fid,'%s',M11_new);
% Close file
fclose(fid);

% Extract timeseries defined in parameters and
% save in file Result.txt The file M11 must be present
% in the same directory
!m11extra MATLAB_run.PRF Result.txt

%%
% Calculate delay
[yy mm dd hh mi ss Accinflow Accoutflow ] = ...
textread('Result.txt','%d -%d -%d %d :%d :%d %f %f',...
'headerlines',5);
MOUSEtimetemp=datenum(yy,mm,dd,hh,mi,ss);
clear yy mm dd hh mi ss


% Find delay based on 50 % quantile of accumulated flow.
ACCInflow(:,i)=Accinflow;
ACCOutflow(:,i)=Accoutflow;
index=find(ACCOutflow(:,i)>=ACCInflow...
(length(ACCInflow),i)/2);
Delay(i)=index(1)*5-90;
% Display status
fprintf('\n Delay calculated \n');
end
```

D.2 Programmes Generated for the Case Study

Several programmes have been developed in order to test the developed flow routing method and the validation method on the case study. The raw MATLAB codes for these programs can be seen

in this section.

```
% Program AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program represents a configuration of the
% developed validation and flow routing methods.
% The methods are configured for the case study.
% Several subprograms are used by this programs.
%
% Subprogram InitialControl.m
% This subprogram conduct the initial control.
%
% Subprogram LocalRedundancy.m
% This subprogram conduct calculate a flow rate based on
% the pressure transducer and calculate the residual.
%
% Subprogram ResidualControl.m
% Test residuals from redundancy subprograms and
% update "ErrorMatrix".
%
% Subprogram ForecastedRedundancy.m
% This subprogram first calulate the delay from the
% water level sensors to the manhole between the sensors.
% Based on the delays the flow rate at the manhole
% is calculated. Afterwars the delay from the manhole
% to the flow sensor is calculated and this delay is used
% to estimate the flowrate at the flow sensor.
% Finally the residuals beween the estimate and the actual
% measurement of the flow rate is calculated.
%
% Subprogram ConductDecision
% This subprogream evaluate the "ErrorMatrix" and
% determine the final outcome
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Set parameters

% Clear all variables
 clear all
 clc
 close all
```

```
% Load Input Data
load('observeddata.mat');

% Initial setpoints - based on manuel inspection
% (Calibration).
h1set=0.015;
h2set=-0.025;

% Correct data according to setpoints
h1=h1+h1set;
h2=h2+h2set;
h=[h1 h2];

% Define threshold to identify outliers. [%]
PercentageThreshhold=15;

% Number of timesteps to run simulation
TotalSim=size(Time,1);

% Allocation of Matrixes
ErrorMatrix=ones(6,5,TotalSim);
QLR=zeros(TotalSim,1);
ResidualLR=QLR;
QFR=QLR;
ResidualFR=QLR;
Delay=zeros(TotalSim,3);
Q_est=QLR;
Q12=QLR;

%%
% Launch Validation

for i=1:TotalSim
    % Test 1-3
    % Conduct initial test based on single sensors,
    % with subprogram InitialControl.m. Test if sensor
    % is on, if it measures a value equal to or
    % less than zero, or a non physical high value
    ErrorMatrix(:,:,i) = InitialControl(Q(i), h1(i), h2(i),...
    vflow(i), hflow(i), gauge(i), ErrorMatrix(:,:,i));

    %%
    % Test 4
    % Local redundancy test
    [ErrorMatrix(:,:,i) ResidualLR(i) QLR(i) VLR(i) ALR(i)] =...
```

```
90        LocalRedundancy(Q(i), hflow(i), ErrorMatrix(:,:,i));
91      % Test Residuals
92      [ErrorMatrix(:,:,i) ThreshholdLR(i)] = ResidualControl...
93      (ResidualLR(i),ErrorMatrix(:,:,i), PercentageThreshhold,...
94      4, [2], Q(i));
95
96      %%
97      % Test 5
98      % Forecasted redundancy test
99      [ResidualFR(i) QFR(i) Delay(i,1:3) ErrorMatrix(:,:,i)...
100     Q12(i)] = ForecastedRedundancy(h(1:i,1:2),...
101     ErrorMatrix(:,:,i), Q(i), Q12(1:i));
102
103     % Test Residuals
104     [ErrorMatrix(:,:,i) ThreshholdFR(i)] = ResidualControl(...
105     ResidualFR, ErrorMatrix(:,:,i), PercentageThreshhold, 5,...
106     [4 5], Q(i));
107
108     % Calculate Total Delays. (For monitoring purpose only)
109     if isnan(Delay(i,3)) == 0;
110        TotalDelay(i,1) = Delay(i,3)+Delay(i-Delay(i,3),1);
111        TotalDelay(i,2) = Delay(i,3)+Delay(i-Delay(i,3),2);
112     end
113
114     %%
115     % Decision making
116     [Q_est(i)] = ConductDecision(ErrorMatrix(:,:,i), QLR(i),...
117     QFR(i), Q(i));
118 end
```

```
1  % Program InitialControl.m
2  % Subprogram for main program: Automatic Validation.m
3  % Programmer: Daniel Brødbæk
4  % Aalborg University
5  % Master Thesis, September 2012 - May 2013
6  %
7  % This program perform the initial control and update
8  % the "ErrorMatrix". The proram is configured for
9  % the Case study.
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 function Control = InitialControl(Q,h1,h2,vflow,hflow,...
13  gauge,Control)
14 % Conduct initial tests
15 %%
```

```
16 % The flow sensor
17
18 % Test if sensor is on/off ie. does data exist
19 if isnan(Q)==1
20    Control(1,1)=0;
21 end
22
23 % Test for non physical value
24 % (Full running approx 190 l/s)
25 % Test if flow is less than or equal to zero
26 if Q<0 || Q>250;
27    Control(1,2)=0;
28 end
29
30 % Test for zero
31 if Q==0;
32    Control(1,3)=0;
33 end
34
35 % Water Depth
36 % Test if sensor is on/off ie. does data exist
37 if isnan(hflow)==1
38    Control(2,1)=0;
39 end
40
41 % Test for non physical value
42 % (Full running pipe 0.6 m)
43 if hflow<0 || hflow>1
44    Control(2,2)=0;
45 end
46 % Test for zero
47 if hflow==0;
48    Control(2,3)=0;
49 end
50
51 % Velocity
52 % Test if sensor is on/off ie. does data exist
53 if isnan(vflow)==1
54    Control(3,1)=0;
55 end
56
57 % Test for non physical value
58 % (Full running pipe approx 0.67 m/s)
59 if vflow<0 || vflow>1
60    Control(3,2)=0;
61 end
```

```
% Test for zero
if vflow==0;
   Control(3,3)=0;
end
%%
% Water level sensor 1

% Test if sensor is on/off ie. does data exist
if isnan(h1)==1
   Control(4,1)=0;
end

% Test for non physical value
% (550 mm should be absolut maximum)
if h1<=0 || h1>0.55;
   Control(4,2)=0;
end

% Test for zero
if h1==0;
   Control(4,3)=0;
end
%%
% Water level sensor 2

% Test if sensor is on/off ie. does data exist
if isnan(h2)==1
   Control(5,1)=0;
end

% Test for non physical value
% (990 mm should be absolut maximum).
if h2<=0 || h2>0.99;
   Control(5,2)=0;
end

% Test for zero
if h2==0;
   Control(5,3)=0;
end

%%
% Rain gauge

% Test if sensor is on/off ie. does data exist
```

```
if isnan(gauge)==1
   Control(6,1)=0;
end

% Test for non physical value.
if gauge<=0 || gauge>33.33;
   Control(6,2)=0;
end

% Test for zero
if gauge==0
   Control(6,3)=0;
end
```

```
% Program LocalRedundancy.m
% Subprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program calculate a flow rate based on the
% pressure transducer and calculate the residual.
% The program is configured for the case study.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Control Residual Q1 V1 A] = LocalRedundancy...
(Q, h, Control)
% Set Errormatrix to NaNs, for measurements which is not
% a target for the validation
Control([2 3 4 5 6],4) = NaN;
%%
% If pressure transducer passed intial tests, then
% conduct local redundancy test.
if sum(Control(2,1:3))==3
    % Calculate flow based on Mannings formula
    theta=acos(1-(2*h/0.6));
    R=0.6*(2*theta-sin(2*theta))/(8*theta);
    % Area corresponding to sedimentation depth is
    % subtracted from the flow area
    A=((0.6^2)/8)*(2*theta-sin(2*theta))-0.005285;
    if A<0
       A=0;
    end
    V1=75*(R^(2/3))*sqrt(0.001);
    Q1=V1*A*1000;
```

```
32
33      % Only conduct test on residuals if flow also
34      % passed initial tests
35      if sum(Control(1,1:3))==3
36         % Calculate residual between measurement
37         % and estimate based on  pressure sensor
38         Residual = Q-Q1;
39      else
40         % If flow havent been estimated, set
41         % ErrorMatrix and residual to NaN
42         Residual = NaN;
43         Control(1,4) = NaN;
44      end
45  else
46     % Set values to NaNs indicating no flow rate
47     % have been estimated
48     Q1 = NaN;
49     V1 = NaN;
50     A = NaN;
51     Residual = NaN;
52     Control(2,4) = NaN;
53  end
```

```
1  % Program ResidualControl.m
2  % Subprogram for main program: AutomaticValidation.m
3  % Programmer: Daniel Brødbæk
4  % Aalborg University
5  % Master Thesis, September 2012 - May 2013
6  %
7  % This program test the residauls calculated in
8  % the redundancy tests and log the performance in the
9  % "ErrorMatrix".The program is configured
10 % for the case study.
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12
13 function [Control Threshold] = ResidualControl(Residual,...
14          Control, percentage, test, Sensors, Q)
15 %%
16 % Test if Residuals was calculated
17 if isnan(Residual)==0
18    Threshold=(percentage*Q)/100;
19    % The minimum threshold is set to 1 l/s.
20    if Threshold<1
21       Threshold=1;
22    end
```

```
    % Test for outlier.
    if abs(Residual)>Threshold
        Control(Sensors,test)=0;
    end
else
    Threshold=NaN;
end
```

```
% Program ForecastedRedundancy.m
% Subprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 – May 2013
%
% The program is configured for the case study.
% The program first calulate the delay from the
% water level sensors to the manhole between the
% sensors. Based on the delays the flow rate at the
% manhole is calculated. Afterwars the delay from
% the manhole to the flow sensor is calculated and
% this delays is used to estimate the flowrate at
% the flow sensor. Finally the residuals beween
% the estimate and the actual measurement
%  of the flow rate is calculated.
% The program use to subprograms.
%
% Subprogram EstimateDelayWaterLevel.m
% This subprogram estimate the delay based on a
% water level.
%
% Subprogram EstimateDelayFlowRate.m
% This program estimate the delay based on a
% flow rate.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Residual Qout Delay Control Q12new] =...
         ForecastedRedundancy(h, Control, Q, Q12)
% Set Errormatrix to NaNs, for measurements which
% is not a target for the test
Control([2 3 4 5 6],5) = NaN;

%%
% Hardcoded constants for delay calculations
Manning = [89 73 77];
I = [0.0034 0.0018 0.0015];
```

```
D = [0.35 0.6 0.6];
Dist = [43.8 57 114.4];

%%
% Calculate delays to manhole
for i=1:2
[Delay(i) Qf(i)] = EstimateDelayWaterLevel(Dist(i),...
h(:,i), D(i), I(i), Manning(i), i);
end

% Calculate flow at manhole
if isnan(Qf(1:2))==[0 0];
    Q12(size(Q12,1),1) = sum(Qf);
    Q12new = sum(Qf);
else
    Q12(size(Q12,1),1) = NaN;
    Q12new = NaN;
end

%%
% Calculate delay to sensor 3
i=3;
[Delay(i) Qout] = EstimateDelayFlowRate(Dist(i),...
Q12, D(i), I(i), Manning(i));

%%
% Residuals
% Only calculate residuals if flow also
% passed initial tests
if sum(Control(1,1:3))==3 && isnan(Qout)==0
    Residual=Q-Qout;
else
    Residual=NaN;
    Control(3,5) = NaN;
end
```

```
% Program EstimateDelayWaterLevel.m
% Subsubprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program estimate the delay based on a water level.
% The program is configured for the case study.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [Delay Q] = EstimateDelayWaterLevel(dist,...
h,d,I,manning,i)
% Find water level to compare with and calculate flow
j=1;
V=0;
A=0;
% Values to handle sedimentation
if i==1
   LowerLimit = 0.001;
   FlowReduction = 0;
else
   LowerLimit = 0.03;
   FlowReduction = 0.005285;
end
% Assume travel time to be maximum 25 minutes
while (dist-((V)*(j-1)*60))>0 && j<size(h,1) && j<25
      % Ensure value is within limits
      if h(length(h)-j)>LowerLimit && h(length(h)-j)<d
         theta=acos(1-(2*h(length(h)-j)/d));
         R=d*(2*theta-sin(2*theta))/(8*theta);
         A=((d^2)/8)*(2*theta-sin(2*theta))-...
         FlowReduction;
         V=manning*(R^(2/3))*sqrt(I);
      end
      j=j+1;
end
% Ensure enough data exist
if j<size(h,1)
   Q=V*A*1000;
   Delay=j-1;
else
   Q=NaN;
   Delay=NaN;
end
```

```
% Program EstimateDelayFlowRate.m
% Subsubprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program estimate the delay based on a flow rate.
% The program is configured for the case study.
% The program use on subprogram.
```

```
%
% Subprogram NewtonIteration.
% The subprogram iterate a water depth based on the
% flow rate by Mannnigs eqaution.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Delay Q] = EstimateDelayFlowRate(dist,Qin,D,...
I,m)
% Find water level to compare with and calculate flow
j=1;
V=0;
h=0.1;
% Assume travel time to be maximum 25 minutes
while (dist-((V)*(j-1)*60))>0 && j<size(Qin,1) && j<25
        % Iterate velocity
        [h V] = NewtonIteration(m, I, D, Qin(size...
        (Qin,1)-j),h);
        j=j+1;
end
% Ensure enough data exist
if j<size(Qin,1)
    Delay = j-1;
    Q = Qin(size(Qin,1)-Delay);
else
    Q = NaN;
    Delay = NaN;
end
```

```
% Program NewtonIteration
% Subsubsubprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program iterate the water depth in a circular
% pipe, where the flow rate is known.
% Stationary and uniform flow is assumed and
% the relation between water depth and flow rate
% is based on Mannnins equation.
% Newtons itereation princip is used.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [h V] = NewtonIteration(m, I, D, Q, h)
% Initial condition
f=1;
```

```
Q=Q/1000;
% Define limits of deviation
while f>0.0001 || f<-0.0001
% Function
f = (1/8)*m*((1/4*(1-sin(2*acos(1-2*h/D))/(2*acos...
(1-2*h/D))))*D)^(2/3)* sqrt(I)*D^2*(2*acos(1-2*h/D)...
-sin(2*acos(1-2*h/D)))-Q;
% The derivative of the function
f1 = (1/48)*m*4^(1/3)*((1/2)*sqrt(2)+(1/2*I)*...
sqrt(2))*D^3*(2*acos(1-2*h/D)-sin(2*acos(1-2*h/D...
)))*(-2*cos(2*acos(1-2*h/D))/(D*sqrt(1-(1-2*h/D)^2)...
*acos(1-2*h/D))+sin(2*acos(1-2*h/D))/(acos(1-2*h/D)...
^2*D*sqrt(1-(1-2*h/D)^2)))/((1 -(1/2)*sin(2*acos(1 -2*...
h/D))/acos(1-2*h/D))*D)^(1/3)+(1/32)*m*4^(1/3)*...
((1 -(1/2)*sin(2*acos(1-2*h/D))/acos(1-2*h/D))*D)...
^(2/3)*((1/2)*sqrt(2)+(1/2*I)*sqrt(2))*D^2*(4/(D*...
sqrt(1-(1-2*h/D)^2))-4*cos(2*acos(1-2*h/D))/(D*...
sqrt(1-(1-2*h/D)^2)));
% Next h value is estimated by newton
% iteration method
h = h-(f/f1);
end
V = m*((1/4*(1 - sin(2*acos(1-2*h/D))/(2*acos(1 -2*...
h/D))))*D)^(2/3)*sqrt(I);
```

```
% Program ConductDecisions.m
% Subprogram for main program: AutomaticValidation.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This progream evaluate the "ErrorMatrix" and
% determine the final outcome. The program is
% configured for the case study.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Q_est] = ConductDecision(Control, QAR,...
QMR, Q)
%%
% Destinct between two main scenarios. Did flow
% sensor passed initial control or not.
% If it did pass then check the result from the
% residual control
if sum(Control(1,1:3))==3
    % If abnormal residuals have been detected in
```

```
   %  both redundancy test flow rate is set to
   % estimate from local redundancy.
   if Control(1,4)==0 && Control(1,5)==0
      Q_est=QAR;
   % If abnormal residuals only have been detected
   % with respect to  local redundancy test,
   % then use measurement.
   elseif Control(1,4)==0 && Control(1,5)==1
      Q_est=Q;
   % If no errors have been detected use measurement
   else
      Q_est=Q;
   end
   % If the flow sensor failed the initial control,
   % but the pressure sensor in the flow sensor passed
   %  then use estimate from local redundancy.
elseif sum(Control(1,1:3))<3 && sum(Control(2,1:3))==3
   Q_est=QAR;
   % In case both velocity and waterdepth measurements
   % failed at the flow sensor then use estimate
   % from Water level sensors
elseif sum(Control(1,1:3))<3 && sum(Control(2,1:3))<3
   Q_est=QMR;
end
```

D.3 Program for Improving Velocity Measurements

This section contains the raw MATLAB code used for improving velocity measurements. The program code is configured for the case study. The program uses one subprogram.

```
% Program KalmanFiltering
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 – May 2013
%
% Configuration of a Kalman filter for improving velocity
% estimates from a flow sensor. The program is
% configured for the case study. The program use one
% subprogram.
%
% Subprogram Kalman
% In this subprogram the Kalman filter algortihm is
% launched based on informations from the main program.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
```

```
% Clear all variables
clear all
clc
close all

load observeddata.mat

% System model matrix
A=[1];

% Measurement model matrix
H=[1];

% Initial conditions
xinitial=[0];

% Covariance of measurement noise
R = 0.008^2;

% Initial conditions
x=[0];
P=0.02^2;

for i=1:size(vflow)
% Set measurement
z=vflow(i);
if isnan(z)==1
    z = 0;
end
% Calculate control input for the Kalman filter
% and the process noise covariance.
if i==1
    DeltaX=[0];
    Q = 0.02^2;
else
    theta=acos(1-(2*hflow(i)/0.6));
    Area(i)=((0.6^2)/8)*(2*theta-sin(2*theta))-...
    0.005285;
    Rd=0.6*(2*theta-sin(2*theta))/(8*theta);
    V1=75*(Rd^(2/3))*sqrt(0.001);
    theta=acos(1-(2*hflow(i-1)/0.6));
    Rd=0.6*(2*theta-sin(2*theta))/(8*theta);
    V2=75*(Rd^(2/3))*sqrt(0.001);
    DeltaX =[V1-V2];
    if  isnan(DeltaX)==1
```

```
            DeltaX=0;
        end

        Q=0.001^2+(((0.02^2 -0.001^2)/0.05)*abs(hflow(i)-...
        hflow(i-1)));
        if isnan(Q)==1
            Q = 0.001^2;
        end
    end

    % Launch Kalman filter
    [x P K] = Kalman(A, H, Q, R, x, P, z, DeltaX);

    % Save Kalman filter estimate of velocity
    Vkalman(i) = x;
    end
    QKalman = Area(:).*Vkalman(:)*1000;
```

```
% Program Kalman
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% This program is a subprogram to the main program
% KalmanFiltering. The program represents the
% linear Kalman filter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x P K] = Kalman(A, H, Q, R, x, P, z,...
DeltaX)
%%
% Forecast part

% Forecast of state variable
xp = A*x+DeltaX;

% update of error covariance
Pp = A*P*A' + Q;

%%
% Calculate Kalman gain
K = Pp*H'*inv(H*Pp*H' + R);

%%
% Analysis part
```

```

% Calculate the analysis estimate
x = xp + K * (z - H*xp);

% Update the error covariance
P = (eye(size(x))-K*H)*Pp;
```

D.4 Programmes for ENKF on a 1D Diffusive Wave Model

In this section the raw MATLAB codes used for the ensemble Kalman filter is presented. The main main program use one subprogram in order to calculate a forecast. This subprogram is also documented in this section.

```
% Program ENKF.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
%
% Program for Ensemble Kalman filter. Use subprogram
% DiffWaveModel.m to model open channel flow.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Set parameters

% Close all figures, clear variables and clear
% commandvindow.
clear all
close all
clc

% Number of ensembles
N = 50;

% Number of Computational nodes
nodes=100;

% Temporal discretisation [s]
dt = 0.01;

% Spatial discretization [m]
dx = 1;

% Width of channel
b=0.5;

```

```
% Mannings number
m = 70;

% Bottom slope
I_0 = 0.005;

% Initial conditions of water level for each ensemble
% Assume a water level of 0.2 m, in the channel.
% The water level is corrupted by random noise,
% following normal distribution with zero mean,
% in all nodes.
% Define standard deviaiton of random noise [m].
noisestd = 0.01;
h_initial = 0.2 + noisestd*randn(N,nodes);

% Define time step between assimilation cycles
simtime = 10;

% Need to define upperboundary (fiktive run-off)
Totalsimtime = 150;
u = [0.13
     0.2
     0.25
     0.3
     0.3
     0.3
     0.25
     0.25
     0.2
     0.13
     0.13
     0.13
     0.13
     0.13
     0.13];

% Assume state variables to be water depth in all
% nodal points.
% Generate h matrix. Assume measurements are water
% level in node 25 and 75
H=zeros(2,nodes);
H(1,25)=1;
H(2,75)=1;

% Preallocation
hensemble = zeros(N,nodes,Totalsimtime/simtime);
```

```
htrue = zeros(Totalsimtime/simtime,nodes);
X_f = htrue';
X_a = X_f;
GAIN = zeros(nodes,size(H,1),Totalsimtime/simtime);
ANALYSIS_STATE = zeros(nodes,N,Totalsimtime/simtime);
FORECAST_STATE = zeros(nodes,N,Totalsimtime/simtime);

%%
% Launch calculations

% Use diffusive wave model to propagate each ensemble
% and then update ensemble before it is propagated
% again. Model 10 seconds forward in time and
% then update with measurements.
for t = 1:Totalsimtime/simtime
    % Simulate the truth model
    if t==1
       [htrue(t,:)] = DiffWaveModel(dx, dt, simtime,...
       nodes, I_0, m, hinitial, u(t)+ 0.05*randn(1,1), b);
    else
       [htrue(t,:)] = DiffWaveModel(dx, dt, simtime,...
        nodes, I_0, m,
    htrue(t-1,:), u(t)+ 0.05*randn(1,1), b);
    end
    % Simulate the noisy measurement. Transpose so true
    % state variables is a column vector.
    z = H * htrue(t,:)'+ 0.001*randn(2,1);

    % Step 1 Forecast
    for i = 1:N
        if t==1
           hin=h_initial(i,:);
        else
           hin=hensemble(i,:,t-1);
        end
          [hensemble(i,:,t)] = DiffWaveModel(dx, dt,...
          simtime, nodes, I_0, m, hin, u(t)+ 0.05*...
          randn(1,1), b);
    end
    % The ensembles matrix is transposed so, state
    % variables for each ensemble is in columns,
    % and then saved in new variable named X_f
    X_f = hensemble(:,:,t)';

    % Calculate mean of ensembles in all nodal points.
    x_f_avg = mean(X_f,2);
```

```
    % Calculate ensemble pertubation matrix
    X_f_per = X_f-repmat(x_f_avg,1,N);

    % Step 2 Kalman gain

    % Create observation ensemble pertubation matrix
    Z_per = 0.001*randn(2,N);
    Z = repmat(z,1,N) + Z_per;

    % Calculate Kalman gain
    K = X_f_per * X_f_per' * H' * inv(H * X_f_per *...
    X_f_per' * H' + Z_per * Z_per');

    % Step 3 Analysis part

    % Update state variables for all ensembles
    X_a = X_f + K*(Z - H * X_f);

    % Calculate mean of ensembles in all nodal points,
    % corresponding to optimal state.
    x_a(:,t) = mean(X_a,2);

    % Use updated ensmebles in next timestep.
    hensemble(:,:,t) = X_a';

    % Save some variables for later inspection.
    GAIN(:,:,t)=K;
    ANALYSIS_STATE(:,:,t) = X_a;
    FORECAST_STATE(:,:,t) = X_f;
end

% Evaluate performance by calculating mean squared
% error between predictions and true value.
MSE = (1/nodes) * sum(((x_a-htrue') * 100).^(2),1);

% Evaluate performance with no data assimilation
for t=1:Totalsimtime/simtime
    if t==1
       hin=ones(N,nodes)*0.2;
    end
   [h_noassi(t,:)] = DiffWaveModel(dx, dt,...
   simtime, nodes, I_0, m, hin, u(t), b);
   hin=h_noassi(t,:);
end
MSE_NoAssimilation = (1/nodes) * sum...
```

```
(((h_noassi'-htrue') * 100).^(2),1);

%%
%Plot figures

fig = figure; hold on
ax=gca;
plot1=plot(simtime:simtime:Totalsimtime,...
MSE_NoAssimilation,'r');
plot2=plot(simtime:simtime:Totalsimtime,...
MSE,'b');
ylabel('MSE [cm^2]')
xlabel('Time [s]')
% Insert Legend
leg=legend([plot1, plot2],...
'MSE no data assimilation','MSE ENKF');
% Set loacation of legend
set(leg,'Location','NorthEast')
% Remove box from legend
set(leg, 'Box', 'off')
posleg=get(leg,'Position');
posleg(2)=posleg(2)-0.02;
posleg(4)=posleg(4)+0.02;
set(leg,'Position',posleg);
hold off

fig2 = figure; hold on
ax=gca;
plot1=plot(htrue(5,:)','r');
plot2=plot(htrue(10,:)','b');
plot3=plot(htrue(15,:)','g');
set(ax,'ylim',[0.15 0.35001])
% Insert Legend
leg=legend([plot1, plot2, plot3],...
't = 50 s','t = 100 s','t = 150 s');

% Set loacation of legend
set(leg,'Location','NorthEast')

% Remove box from legend
set(leg, 'Box', 'off')

ylabel('Water depth [m]')
xlabel('Cell index')
hold off

```

```
fig3 = figure; hold on
ax=gca;
plot1=plot(ANALYSIS_STATE(:,:,5));
plot2=plot(ANALYSIS_STATE(:,:,10));
plot3=plot(ANALYSIS_STATE(:,:,15));
set(ax,'ylim',[0.15 0.35001])
ylabel('Water depth [m]')
xlabel('Cell index')
hold off

fig4 = figure; hold on
ax=gca;
plot1=plot(x_a(:,15)','k');
plot2=plot(h_noassi(15,:)','m');
plot3=plot(htrue(15,:)','g');
set(ax,'ylim',[0.2 0.28])
ylabel('Water depth [m]')
xlabel('Cell index')
% Insert Legend
leg=legend([plot1, plot2, plot3],...
'ENKF','No data assimilation', 'True water depth');
% Set loacation of legend
set(leg,'Location','NorthEast')
% Remove box from legend
set(leg, 'Box', 'off')
posleg=get(leg,'Position');
posleg(2)=posleg(2)-0.02;
posleg(4)=posleg(4)+0.02;
set(leg,'Position',posleg);
hold off
```

```
% Program DiffWaveModel.m
% Programmer: Daniel Brødbæk
% Aalborg University
% Master Thesis, September 2012 - May 2013
% A diffusive wave model, based on the continuity
% equation and Mannings equation. Assume rectangular
% channel, constant geometry and constant bottom slope

%
% Input parameters:
% dx = spatial discretisation [m]
% dt = temporal discretisation [s]
% nodes = number of computational nodes
% I_0 = bottomslope [m/m]
```

```
% m = Mannings number [m^(1/3/s)]
% b = width of channel
% simtime = simulationtime [s]
% incon = vector with initial conditions in all nodes
% upboundary = vector with upper boundary condition in
% all timesteps. The upper bouundary should be defined
% as a flowrate [m^3/s]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [hout] = DiffWaveModel(dx, dt, simtime,...
         nodes, I_0, m, incon, upboundary, b)

% Preallocation
h = zeros(simtime/dt,nodes);
Q = zeros(simtime/dt,nodes);

% Calculate z-levels at nodal points
z(1)=0;
for k=2:nodes+1
    z(k)=z(k-1)-I_0*dx;
end

% Calculate flow corresponding to initial
% conditions of water level.
% This is necesarry for a hotstart of the model.
% Flow is calculated inbetween h nodes
for j=1:nodes
    if j==nodes
       % Applying boundary condition
       Slope=(incon(j)+z(j)-(incon(j)+z(j+1)))/dx;
       R = (((incon(j)+incon(j))/2)/(1+2*(((incon(j)+...
       incon(j))/2)/b)))^(2/3);
       if Slope<0
          posSlope=-1*Slope;
          Q_initial(j) = -m * R  * ((b*((incon(j)+...
          incon(j))/2)))* sqrt(posSlope);
       else
          Q_initial(j) = m * R * ((b*((incon(j)+...
          incon(j))/2))) * sqrt(Slope);
       end
    else
       Slope=(incon(j)+z(j)-(incon(j+1)+z(j+1)))/dx;
       R = (((incon(j)+incon(j+1))/2)/(1+2*...
       (((incon(j)+incon(j+1))/2)/b)))^(2/3);
       if Slope<0
          posSlope=-1*Slope;
          Q_initial(j) = -m * R * ((b*((incon(j)+...
```

```
            incon(j+1))/2)))* (posSlope)^(1/2);
        else
            Q_initial(j) = m * R * ((b*((incon(j)+...
            incon(j+1))/2)))* (Slope)^(1/2);
        end
    end
end

% Launch model to estimate change in time.
% i = time index
% j = spatial index
for i=1:simtime/dt
    for j=1:nodes
        % Calculate water level in all nodes. The water
        % level is estimated based on flow at
        % last timestep. If i = 1 use initial conditions.
        if i==1
            if j==1
                h(i,j) = ((upboundary-Q_initial(j))/dx*...
                b)* dt + incon(j);
            else
                h(i,j) = ((Q_initial(j-1)-...
                Q_initial(j))/dx*b)* dt + incon(j);
            end
        elseif j==1
            h(i,j) = ((upboundary - Q(i-1,j))/dx*b) *...
            dt + h(i-1,j);
        else
            h(i,j) = ((Q(i-1,j-1) - Q(i-1,j))/dx*b) *...
            dt + h(i-1,j);
        end
    end
    % Calculate flow inbetween all nodes
    for j=1:nodes
        if j==nodes
            % Applying boundary condition
            Slope=(h(i,j)+z(j)-(h(i,j)+z(j+1)))/dx;
            R = (((h(i,j)+(h(i,j))/2)/(1+2*(((h(i,j)+...
            (h(i,j))/2)/b))))^(2/3));
            if Slope<0
                posSlope=-1*Slope;
                Q(i,j) = -m * R * ((b*((h(i,j)+...
                h(i,j))/2))) * (posSlope)^(1/2);
            else
                Q(i,j) = m * R * ((b*((h(i,j)+h(i,j))/2)))...
                * (Slope)^(1/2);
```

```
            end
        else
            Slope=(h(i,j)+z(j)-(h(i,j+1)+z(j+1)))/dx;
            R = ((h(i,j)+(h(i,j+1))/2)/(1+2*(((h(i,j+1)+...
            (h(i,j))/2)/b))))^(2/3);
            if Slope<0
                posSlope=-1*Slope;
                Q(i,j) = -m * R * ((b*((h(i,j)+h(i,j+1))/2)))...
                * (posSlope)^(1/2);
            else
                Q(i,j) = m * R * ((b*((h(i,j)+h(i,j+1))/2)))...
                * (Slope)^(1/2);
            end
        end
    end
end

hout = h(i,:);
```